XB1 User's Manual

Sun Microelectronics

Contents

List of Figures

List of Tables

Overview 1

1.1 Introduction

This document describes the XB1[1] crossbar switch. The XB1 acts as the bridge between three system buses. Only one of the buses is dedicated to a particular function, that of interfacing to memory, while the other two are general-purpose buses. Suggested uses are for a processor bus, a memory data bus, and an input-output (I/O) bus in SPARC Technology Business' reference uni-processor design for the UltraSPARC™. Owing to different system requirements, these buses are different widths. To keep the cost economical and provide implementation flexibility, the part is sliced such that multiple chips are required to build a system.

The XB1 consists of one four-bit port, one eight-bit port, and one 16-bit port.

The A[2] port (16 bit) is intended for use as a memory interface. The B port (eight bit) is intended for use as a microprocessor interface. The C port (four bit) is intended for use as an I/O interface. The B and C ports have approximately 10-ohms damping resistor on the outputs. All ports are controlled by state machines which sequence the filling and the transferring of data into and from internal registers.

The memory port can handle only block transfers. When MRB_CTRL is asserted, data will be transferred from the memory bus into the memory's read registers. The memory read commands will result in transferring data from the memory registers to the bus specified by the command. The memory write command will result in transferring data from the processor bus, or the I/O bus, to the memory

1. In previous documentation, the XB1 was called STP2230SOP, and was code named the BMX.
2. In previous documentation, the A_BUS, B_BUS, and C_BUS were referred to as M_BUS, P_BUS, and I_BUS respectively.

write registers. When MWB_CTRL is asserted, data will be transferred from the memory write registers to the memory bus. The memory port allows wrapping management for both processor accesses and I/O accesses. The wrapping management is only applicable to data transfers involving memory read commands, (see Chapter 4, "Operational Details").

If the control signal MRB_CTRL and a memory read command are received at the same rising edge of the clock, the first memory byte or nibble, in case of a memory to I/O transfer, will bypass the memory registers and appear on the appropriate bus.

The data transfers are controlled by commands which are delivered to the select pins, SEL3, SEL2, SEL1, and SEL0. Input data to any ports will be treated and arranged assuming "big endian" ordering. The highest count of bits will always refer to the most significant bit. The most significant bit for the memory bus will appear on pin A15, for the processor bus will appear on pin B7 and for the I/O bus will appear on pin C3. The chip will always allocate the resources required by the last command asserted. An external controller is needed to resolve resource conflicts that could arise from two simultaneous commands. All commands, except the idle and reset commands, will initiate data transfers.

The assertion of the RESET command will set all buffers to zero and place all ports in the high-impedance state. The assertion of the TEST command, will cause the processor bus data to be copied to the memory bus and to the I/O bus.

The IDLE command is the default command on the chip. During the IDLE command, any other command that requires multiple clock cycles will continue execution until completion.

1.2 Features

The XB1 is a crossbar switch with the following features:

- Three-port crossbar
- Decoupled memory port - loading and unloading of memory data can take place in parallel with other operations
- Burst transfers operate on four bytes of data per slice
- Power-up safe buses - all buses power up tristated so there will be no bus contention with other parts which may be on the buses
- Implemented in 0.8-micron Bi-CMOS and housed in a 48-pin TSSOP package (also called DGG or "Shrink wide-bus")

1.3 Block Diagram

There are four major blocks which make up the internal structure of the XB1:

- memory data interface block
- I/O data interface block
- processor data interface block
- command decoder block

This device performs a data-switching function among any of three buses. All port-to-port transfers are registered at the input bus, and at the output bus. Due to the possible simultaneous assertion of the memory read control (MRB_CTRL), and a memory read command, a bypass multiplexer allows data to flow from the A_BUS to the B_BUS output register without an additional input register delay. This minimizes latency for memory accesses.

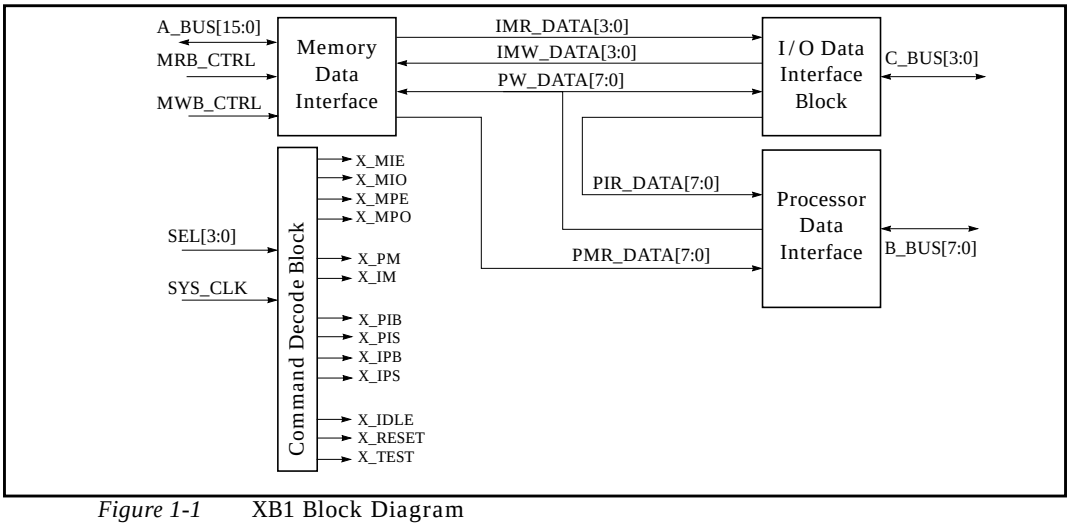

Figure 1-1 XB1 Block Diagram

Sections which follow provide more detail on the internal structure of the various blocks. The memory data interface block is made up of two pieces: the memory read data interface and the memory write data interface.

1.3.1 Memory Read Data Interface

The memory read data interface consists of independent loading and unloading logic. Read data is input through the A_BUS under control of the MRB_CTRL signal. Unloading proceeds based on any of the X_M* commands.

Note: The symbol "*" is used as a wild card. It can symbolize a processor [P], or input/output [I], or it can be odd or even. It can also symbolize either: X_MPE, X_MPO, X_MIE, and X_MIO, or possible combinations thereof.

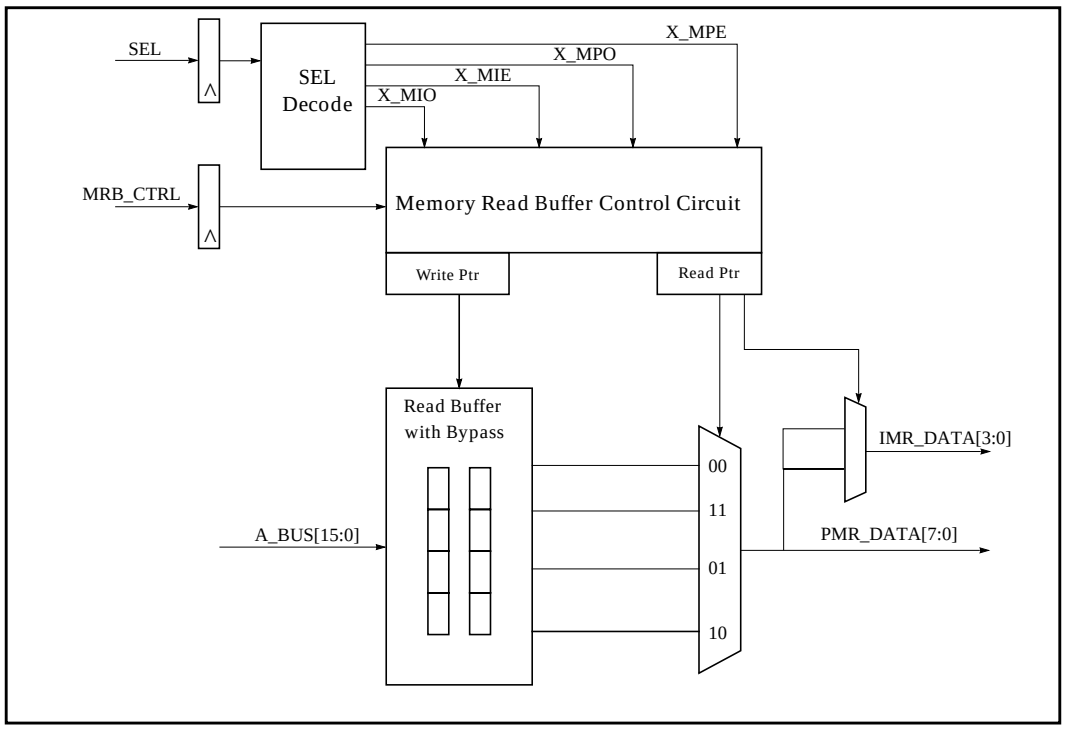

Figure 1-2 XB1 Memory Read Data Interface Block Diagram

1.3.2 Memory Write Data Interface

For memory writes, data sinking from the source bus, and sourcing onto the A_BUS are decoupled. MWB_CTRL determines when valid data which has been loaded through an X_[I,P]M command and is transferred out to the A_BUS.

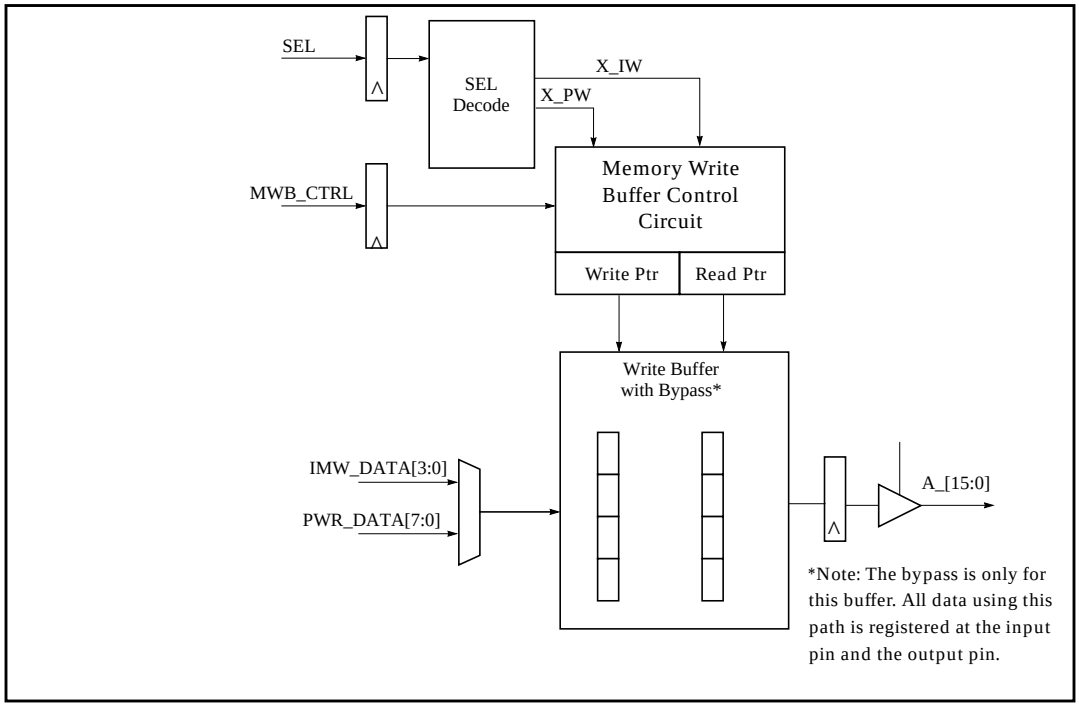

Figure 1-3 Memory Write Data Interface Block Diagram

1.3.3 I/O Bus Data Interface

I/O reads and writes are controlled by the I/O bus data interface. All transfers between the C_BUS and B_BUS follow a deterministic timing, they are always coupled.

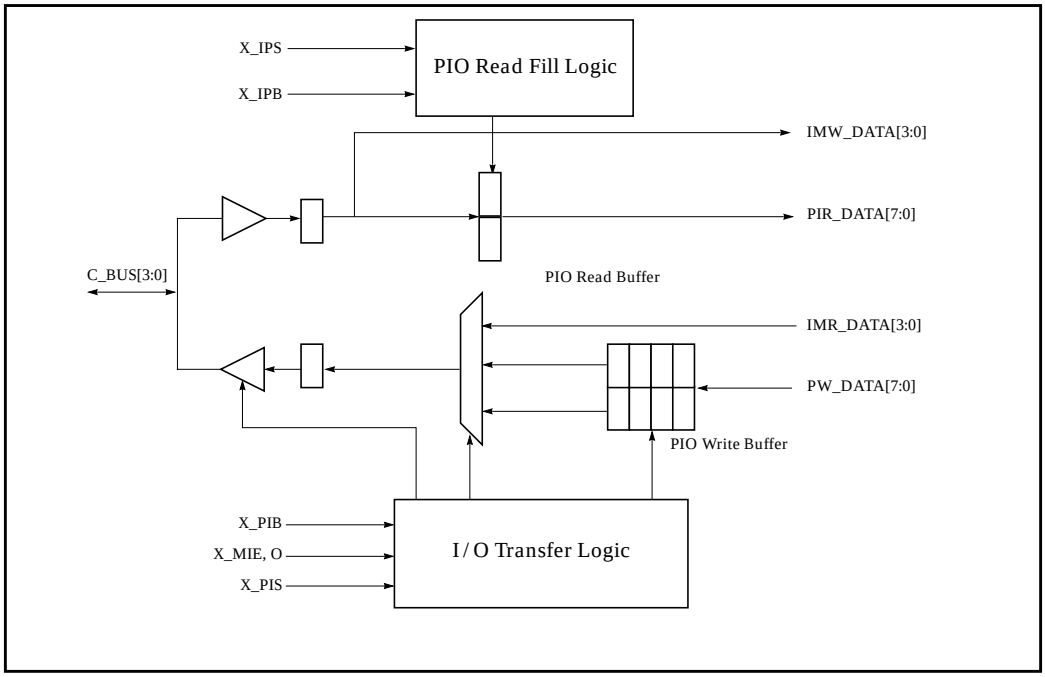

Figure 1-4 XB1 I/O Bus Data Interface

1.3.4 Processor Data Bus Interface

This block controls all processor port activity.

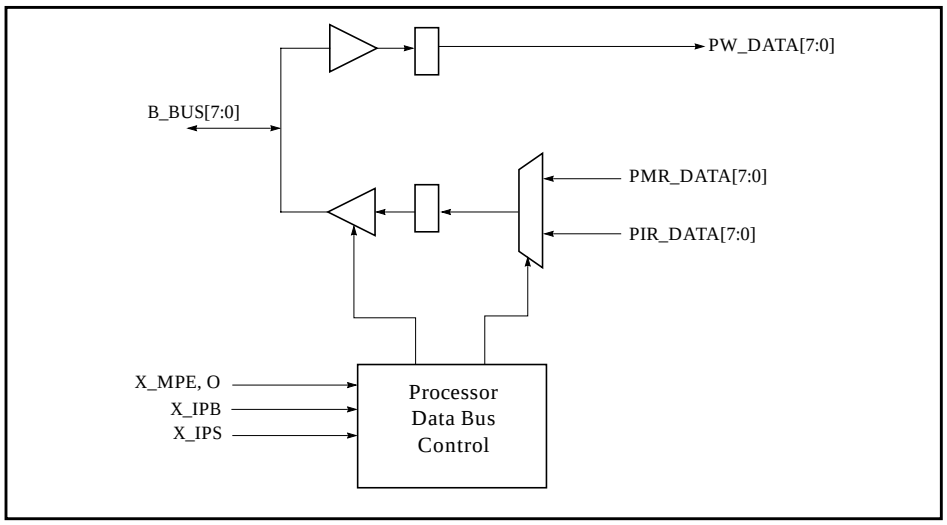

Figure 1-5 Processor Data Interface Block

Pin List and Description 2

This section describes the XB1 pins.

2.1 Summary

See the “Crossbar Switch (XB1) Data Sheet” for pinout information (mechanical and electrical). Table 2-1, "XB1 Signal List Summary," identifies the XB1 signal names.

Table 2-1 XB1 Signal List Summary

Signal Name	I/O	No. of Pins	Description	I/F Type
A[15:0]	I/O	16	Bidirecitonal data from memory bus. Must withstand 5-volt inputs.	LVTTL, 5-volt tolerant
B[7:0]	I/O	8	Input/output bus, bidirectional, registered	LVTTL
C[3:0]	I/O	4	Processor bus, bidirectional, registered	LVTTL
MRB_CRTL	I	1	Memory read buffer control	LVTTL
MWB_CRTL	I	1	Memory write buffer control	LVTTL
SYS_CLK	I	2	Balanced clock signals	3.3-volt PECL
SEL[3:0]	I	4	Command input signals	LVTTL
Total signals		36		
Power and grounds		12		

Table 2-1 XB1 Signal List Summary

Signal Name	I/O	No. of Pins	Description	I/F Type
Total pins:		48		

2.2 Detailed Description

2.2.1 A[15:0]

This port has a two-word (16 bits per word) read-data buffer that is loaded under the control of a specific pin on the crossbar (MRB_CTRL). There is no input register on this port as memory reads are of an asynchronous nature, and the extra cycle required would increase memory latency. All data either goes into the memory read buffer, or through a bypass path directly into the B_BUS output register.

There is also two-word (16 bits per word) write-data buffer. This buffer may be used to retrieve and hold a processor's write-back data even though the memory chips aren't yet ready to receive it. The write data buffer is emptied under the explicit control of the external signal (MWB_CTRL).

The memory port can transfer data to/from either the processor port or the I/O port. The memory port can handle only block transfers, however, it does allow wrapping management for both processor accesses and I/O accesses. See "Wrapping Management" in Section 4.1 of Chapter 4 "Operational Details," for a discussion.

2.2.2 B[7:0]

This port contains no buffers, just multiplexing and I/O registers. The processor port can transfer data to/from any other port, but not itself.

2.2.3 C[3:0]

This port contains eight nibble-read and write buffers. The read and write buffers are used to buffer data transfers between the processor port and the I/O port (PIO transfers). These buffers can handle both single-byte and four-byte transfers.

This port can also transfer four-byte block data to and from the memory port buffers for DVMA accesses.

2.2.4 SEL[3:0]

Data is transferred by commands which are delivered over the SEL control port. Each command has a unique encoding and acts as a specific instruction to the XB1. Except for idle, test, and reset, all XB1 commands initiate a transfer. A summary of the commands is given in the table below. See Chapter 3 "Command Description," for a detailed discussion of the commands, their function, and their timing.

Table 2-2 XB1 Commands

Memory read commands:	X_MPE, X_MPO, X_MIE, X_MIO
Memory write commands:	X_PM, X_IM
Port-to-port commands:	X_PIS, X_IPS, X_PIB, X_IPB
Miscellaneous commands:	X_IDLE, X_IDLE1, X_IDLE2, X_TEST, X_TEST2, X_RESET

XB1 commands are asserted for one cycle to become active. The command is active for as long as it takes to perform the implied transfer. The XB1 can handle multiple overlapping commands (for example, a second command may be issued before the first command completes). The amount of overlap allowed is described in Chapter 3 "Command Description," Section 3.6 "Command Restrictions." An external controller is responsible for resolving resource conflicts that could arise from two simultaneous commands. The resources required by the last command to be asserted are always those allocated by the XB1.

The reset command is asserted after powerup. The X_IDLE command is asserted while any other command is being completed and when no data transfer command is being performed. It is possible for two or more commands to be active simultaneously. They must be initiated sequentially, and must not cause XB1 resource conflicts.

2.2.5 SYS_CLK

The XB1 takes in a differential Pseudo Emitter Coupled Logic (PECL) clock and uses it for all timing references.

Note: With clocks non-operational, the XB1 is guaranteed to come up with output pins tristated. Additionally, operating the part with clocks disabled will not result in damage to the part.

2.2.6 Power and Ground

Both 5-V and 3.3-V power is supplied to this part. Memory port inputs must be 5-V tolerant while all other ports are strictly 3.3 V. All output buffers, including the A_BUS, drive to 3.3 volts.

Note: It is permissible to bring up 5-V and 3.3-V power in any order although proper operation is only guaranteed if both supplies are at the proper voltage. Outputs must be tristated regardless of the order (or presence) of both supplies.

Command Description 3

3.1 Overview

Transfer of data from a port to any other port requires issuing an XB1 command. Like all ports except memory, the command port is registered, so commands take affect the cycle after they are issued. Commands must be issued for *only* one cycle, but will remain active until the implied transfer is completed. An external controller is responsible for resolving resource conflicts that could arise from issuing commands which have overlapping lifetimes and must delay commands from issuing until their resources are available. This is described in more detail in Section 3.6, "Command Restrictions,"

Each command has a unique encoding and acts as a specific instruction to the XB1. Except for idle, test, and reset, all XB1 commands initiate a transfer. The nomenclature for each command is given below.

Memory read commands (X_MIO, X_MIE, X_MPO, and X_MPE) require that there be data in the Memory Read Buffer (MRB). Consequently, the MRB_CTRL signal must have been asserted at least once prior to or coincident with the issuance of the command. The command cannot complete until four bytes have been transferred, so a second MRB_CRTL assertion must occur before the command can complete.

3.1.1 XB1 Command Nomenclature

x_[Source designator][Destination designator][Qualifier]

Where:

Source, Destination designators:

P = B_BUS (Processor)
I = C_BUS (I/O
M = A_BUS (Memory)

Qualifier:

E=Even alignment (to M as source transfers only)

O = Odd alignment (to M as source transfers only)

B = Block transfer (non-A_BUS transfers)

S = Single transfer (non-A_BUS transfers)

Example:

X_MPE is a transfer from the A_BUS to the B_BUS with even alignment.

The reset command is asserted after powerup. The X_IDLE command is asserted while any other command is being completed and when no data transfer command is being performed. It is possible for two or more commands to be active simultaneously. They must be initiated sequentially, and must not cause XB1 resource conflicts.

For XB1 command encoding, see “XB1 Command Encoding” in Table 3-1.

Table 3-1 XB1 Command Encoding

BMX_CMD[3:0]	Command Name
0000	X_IDLE
0001	X_MIO
0010	X_MIE
0011	X_MPO
0100	X_MPE
0101	X_PM

BMX_CMD[3:0]	Command Name
0110	X_IM
0111	X_PIS
1000	X_PIB
1001	X_IPS
1010	X_IPB
1011	X_IDLE1
1100	X_IDLE2
1101	X_TEST2
1110	X_TEST
1111	X_RESET

Note:
BMX_CMD[3] denotes SEL3
BMX_CMD[2] denotes SEL2
BMX_CMD[1] denotes SEL1
BMX_CMD[0] denotes SEL0

The sections which follow describe each of the commands in detail. They are grouped according to the type of transaction performed.

3.2 Memory Read Buffer Commands

3.2.1 X_MI[E,O]

Transfer two 16-bit words of data from the A_BUS to the C_BUS. Loading of the data into the XB1 is under the control of MRB_CTRL. Internally, the XB1 must keep track of how many times MRB_CTRL has been asserted thus far, and ensure that its internal data selector stalls when valid data is not yet available. Data ordering is specified as per the wrapping management indicated by this command. See Chapter 4 “Operational Details," Section 4.1, "Wrapping Management," for details on wrapping management.

This command blocks the C_BUS until the cycle is completed. The minimum duration for data on the C_BUS will be for eight clock cycles and will be extended by the buffer state machines until valid data is supplied through the use of the

MRB_CTRL signal. At least one MRB_CTRL must be asserted prior to or coincident with this command. See Chapter 4 “Operational Details," Section 4.2, "Memory Read Buffer Operation," for a discussion of buffer overwriting and pipelining of this command.

Figure 3-1, "X_ MIE Basic Timing," shows the basic X_MIE timing. Note the data output timing with respect to the data input timing. This is an example where input data bypasses the memory read buffer and goes directly into the output register.

Timing for X_MIO is identical, the only difference is data delivery as discussed in Section 4.1, "Wrapping Management," of Chapter 4. The second MRB_CTRL pulse is shown at the latest possible time to insure contiguous C_BUS data transfer.

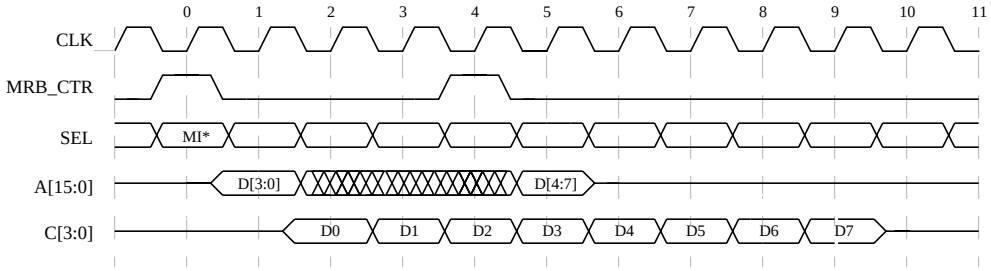

Figure 3-1 X_ MIE Basic Timing

If a second MRB_CTRL has not been received in time for contiguous data delivery, the bus will remain enabled and the data is a “don’t care” until after the second MRB_CTRL is received, see Figure 3-3, "X_MPE Basic Timing," for an example.

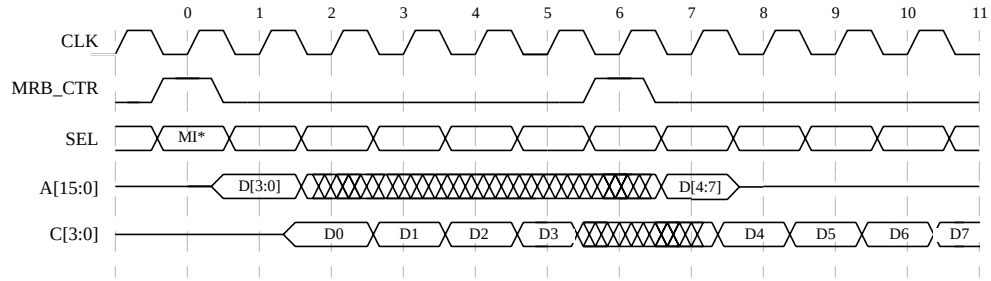

Figure 3-2 X_ MIE Timing Controlled by MRB_CTRL

3.2.2 X_MP[E,O]

Transfer two 16-bit words of data from the A_BUS to the B_BUS. The B_BUS will be busy for four cycles to transfer the A_BUS data. Loading of the data into the XB1 is under the control of MRB_CTRL. Internally, the XB1 must keep track of how many valid transfers have been issued thus far and ensure that its internal data selector stalls when valid data is not yet available. Data ordering is specified as per the wrapping management indicated by this command. See Chapter 4 “Operational Details," Section 4.1, "Wrapping Management," for further details about wrapping management.

This command blocks the B_BUS until the cycle is completed. The minimum duration for data on the B_BUS will be for four clock cycles and will be extended by the buffer state machines until valid data is supplied through the use of the MRB_CTRL signal. At least one MRB_CTRL must be asserted prior to, or coincident with this command. See Chapter 4 “Operational Details," Section 4.2, "Memory Read Buffer Operation," for details of buffer overwriting and pipelining of this command.

Figure 3-3, "X_MPE Basic Timing," illustrates the basic timing and indicates the latest time that a second MRB_CTRL can be asserted and still maintain contiguous data output on the B_BUS.

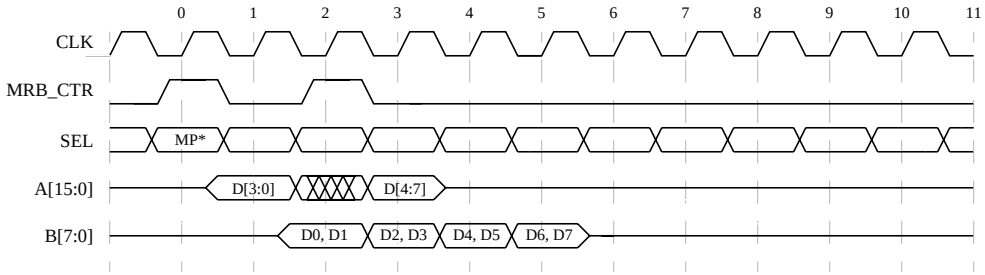

Figure 3-3 X_MPE Basic Timing

Figure 3-4, "X_MPE Timing Controlled by MRB_CTRL," illustrates the case where the B_BUS data delivery (for the second pair of cycles) is delayed by MRB_CTRL.

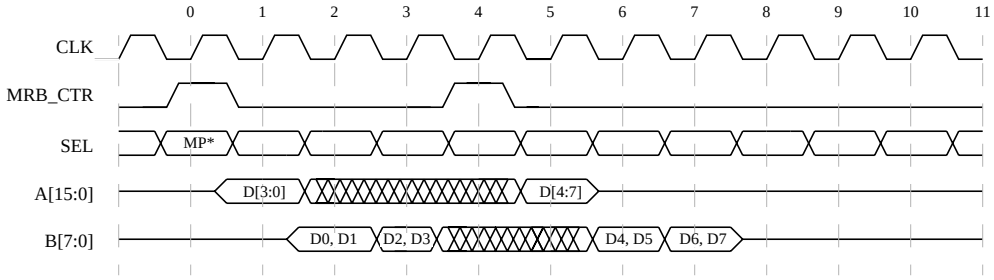

Figure 3-4 X_MPE Timing Controlled by MRB_CTRL

3.3 Memory Write (MW) Buffer Commands

3.3.1 X_PM

Transfer four consecutive bytes of data from the B_BUS to the A_BUS. The A_BUS transfer will be in two 16-bit words, under the explicit control of the MWB_CTRL signal. A subsequent command which uses the write buffer cannot be issued until there is space in the write buffer. See Chapter 4, Section 4.3, "Memory Write Buffer Operation," for the minimum back-to-back transfer timing. Figure 3-5, "X_PM Basic Timing," illustrates the command X_PM basic timing.

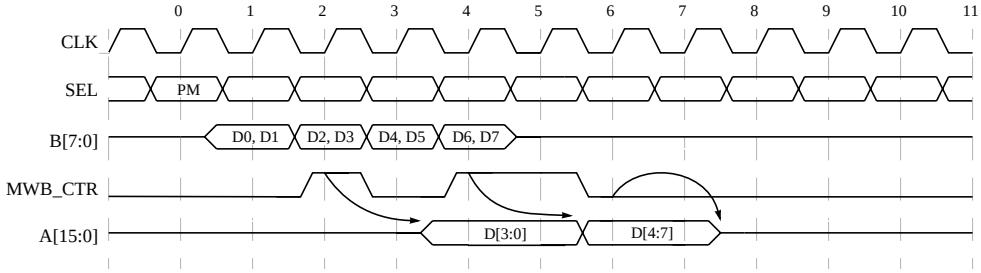

Figure 3-5 X_PM Basic Timing

3.3.2 X_IM

Transfer eight consecutive nibbles of data from the C_BUS to the A_BUS. The A_BUS transfer will be in two 16-bit words, and under the explicit control of the MWB_CTRL signal. A subsequent command which uses the write buffer cannot be issued until there is space available in the write buffer.

Figure 3-6, "X_IM Basic Timing," shows the earliest that MWB_CTRL can be asserted relative to C_BUS data. In order to minimize bus utilization on the A_BUS, the first MWB_CTRL can be delayed.

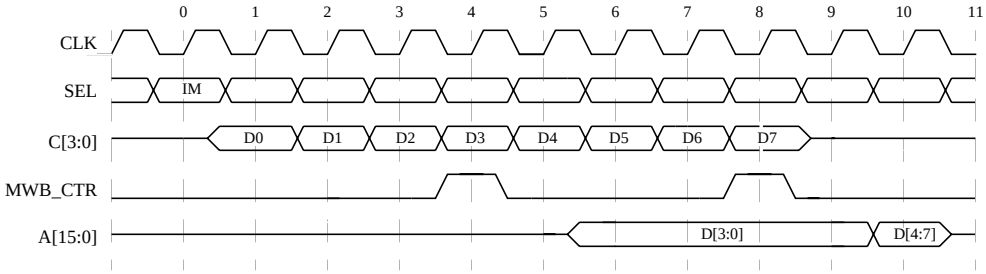

Figure 3-6 X_IM Basic Timing

3.4 PIO Commands

3.4.1 X_PIS

Transfer from B_BUS to the C_BUS one byte of data. The data is assumed valid one cycle following the assertion of XB1_CMD[3:0], and the C_BUS is enabled with the data two cycles following that. This command may be pipelined. X_PIS commands can only be issued at most every other cycle. The C_BUS will remain enabled until all data are transferred. Figure 3-7, "X_PIS Basic Timing," illustrates command X_PIS basic timing.

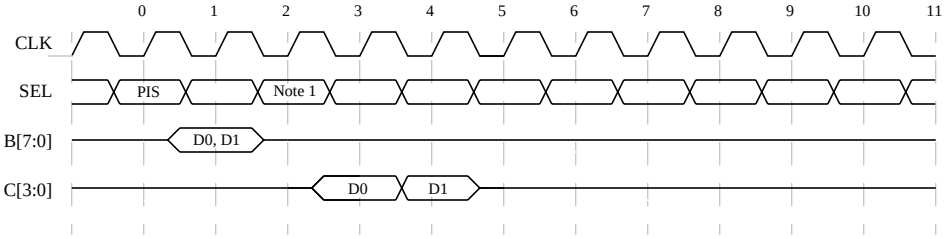

Figure 3-7 X_PIS Basic Timing

Note: This is the earliest that another X_PIS command can be issued.

For the pipeline timing of this command, see Section 4.4 , "I/O Write Register Operation," in Chapter 4.

3.4.2 X_PIB

Transfer from the B_BUS to the C_BUS four bytes of data (a block). As with all other commands, there must be a dead cycle on the source (B_BUS) and destination (C_BUS). The data / enable timing is similar to that of X_PIS. Figure 3-8, "X_PIB Basic Timing," illustrates the command X_PIB basic timing.

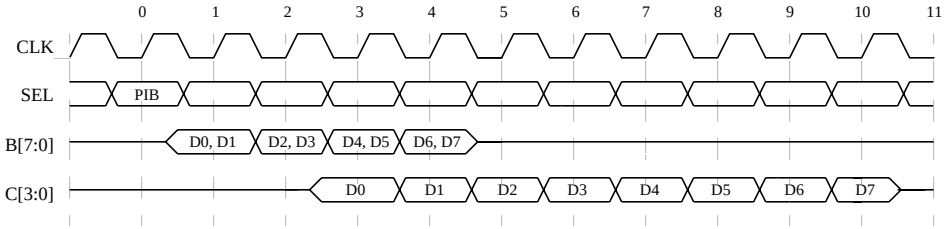

Figure 3-8 X_PIB Basic Timing

3.4.3 X_IPS

Transfer two nibbles of C_BUS data to the B_BUS. This will be one byte on the B_BUS. There are no data first-in / first-out (FIFO) restrictions with this command as the C_BUS limits the speed at which the transfer can take place. See the illustration of the command X_IPS basic timing, in Figure 3-9.

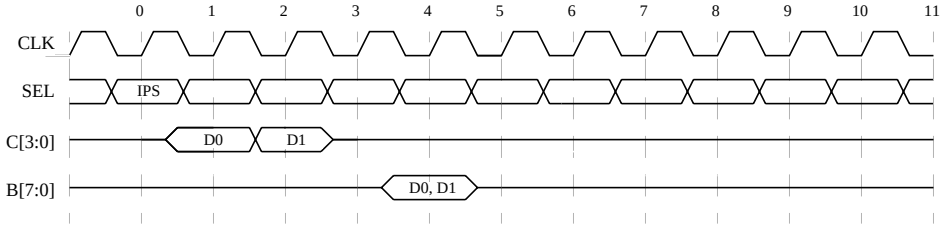

Figure 3-9 X_IPS Basic Timing

3.4.4 X_IPB

Transfer eight nibbles of C_BUS data to the B_BUS. There are no FIFO restrictions with this command as the C_BUS limits the speed at which the transfer can take place. Note that data is declared to be valid for only one cycle for each byte output on the B_BUS. During the times in between, it is a “don’t care.” Figure 3-10, "X_IPB Basic Timing," illustrates the command X_IPB basic timing.

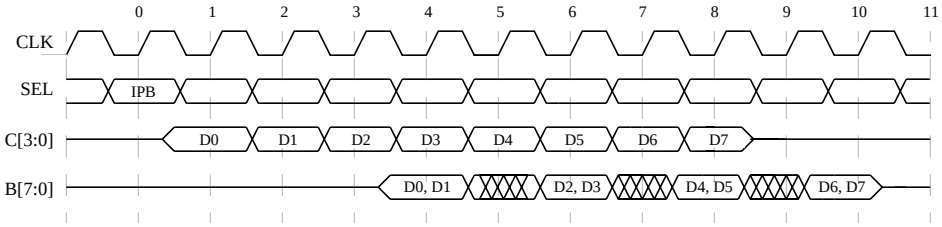

Figure 3-10 X_IPB Basic Timing

3.5 Other Commands

3.5.1 X_IDLE, X_IDLE1, X_IDLE2, X_IDLE3

Idle commands are the default command issued by a controller to the XB1. They result in no transfer of data. Multiple idle commands have been assigned to make sure that all command encodings are specified.

3.5.2 X_TEST

Causes B_BUS data to be copied onto A_BUS and C_BUS. This command is supplied so that all nodes are visible for board test. Without this command, it would be necessary to implement JTAG for the part. Note that for the A_BUS, the B_BUS data is replicated twice due to the port width difference. The part must be reset prior to issuing this command.

The behavior of this command is illustrated in Figure 3-11, "X_TEST Basic Timing," Note that the cycle following the test command is a “don’t care,” it is ignored. This permits X_TEST commands to be issued continuously.

It is mandatory that the last X_TEST command be followed by at least five X_IDLE commands in order to insure proper operation. This clears out any internal state and insures that no following command can cause a bus conflict.

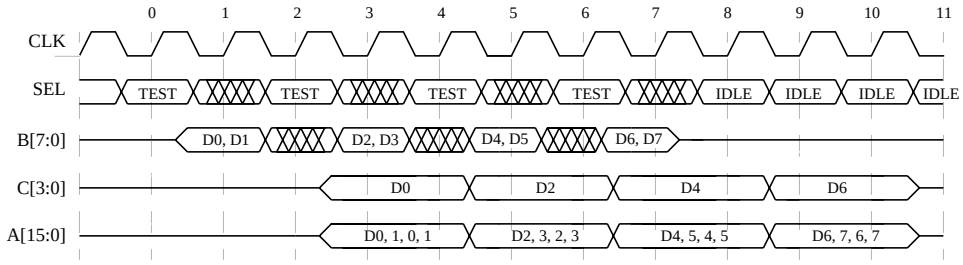

Figure 3-11 X_TEST Basic Timing

3.5.3 X_TEST2

The X_Test2 command causes A_BUS to be copied onto C_BUS and B_BUS. The difference between this command and the previous command is in which nibble of the A_BUS gets copied on the B_BUS. In the previous command, the MS nibble gets copied to the B_BUS. With this command, the LS nibble gets copied to the C_BUS. It is mandatory that the (last) X_TEST2 command be followed by at least five X_IDLE commands in order to ensure proper operation. This clears out any internal state and ensures that no following command can cause a bus conflict.

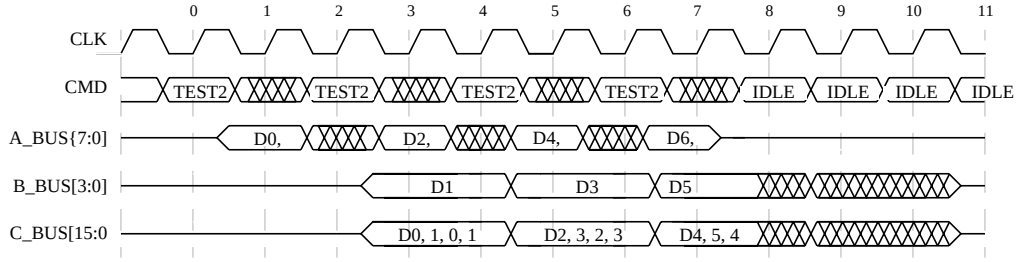

Figure 3-12 X_TEST2 Basic Timing

3.5.4 X_RESET

The X_RESET initializes all counters to zero, clears all buffer flags, and places all of the ports into high-impedance state. Operations currently in progress are terminated immediately (for example, they do not run to completion). This command must be followed by at least one X_IDLE command before any of the data transfer commands are given. Figure 3-13, "X_RESET Basic Timing," illustrates command X_RESET basic timing.

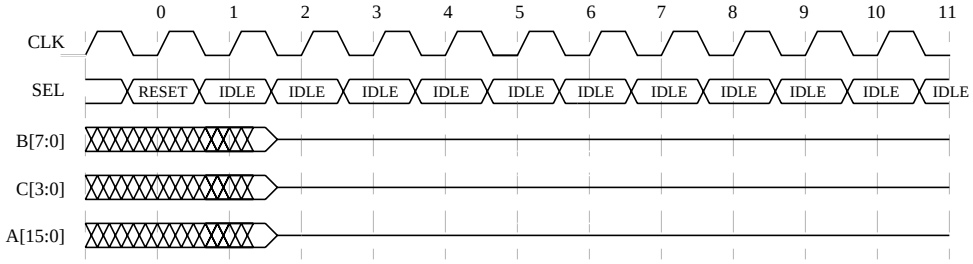

Figure 3-13 X_RESET Basic Timing

Note: MRB should not be asserted for four cycles after issuing reset. Additionally, MRB_CTRL should *NOT* be asserted while reset is active.

3.6 Command Restrictions

Table 3-2, "Command Issuing Restrictions," specifies the earliest that a given command can be issued relative to the previous command which may still be active. For any of the first commands listed, a subsequent command can be issued N cycles later, where N is the value in the table. A value of one means that the command can be issued in the next cycle.

Note: All transactions except for X_PIS result in at least one dead cycle between transfers on all buses.

For cases where two numbers are listed (n/m), the restriction on issuing the second command is n cycles after the first command, or m cycles after (or before if negative) the appropriate control signal (MRB_CTRL or MWB_CTRL) whichever is *LATER*. For MRB_CTRL, the reference is to its second assertion. For MWB_CTRL, the reference is to the cycle where the second MWB_CTRL is de-asserted. Values in the table referenced against MWB_CTRL are negative indicating that the command can precede the control signal by that many cycles. So for example, a PIB command which follows an MIE command can come eight cycles after the PIB if the second MRB_CTRL is early enough, otherwise, it must come four cycles after the MRB_CTRL. Likewise, an IM command following a PM command must come at least four cycles later, and if the MWB_CTRL is either delayed or remains asserted for long enough, the IM command could come four cycles before the de-asserting edge of the second MWB_CTRL.

Table 3-2 Command Issuing Restrictions

First Transaction	Second Transaction: MI*	MP*	PM	IM	PIS	IPS	PIB	IPB
MI*[1]	9/5	8/4	1	10/6	8/4	10/6	8/4	10/6
MP*[1]	4/2	5/3	6/4	1	6/4	3/1	6/4	3/1
PM[2]	1[3]	4[1]	5/–2	4/–4	5	2	5	2
IM[2]	8[1]	1[1]	8/–2	9/–2	7	9	7	9
PIS	4	1	2	5	2	5	3	5
IPS	2	4	5	3	5	3	5	3
PIB	10	4	5	11	9	11	9	11
IPB	8	10	11	9	11	9	11	9

1. If a second number is listed, it is with respect to the assertion of the second MRB_CTRL.
2. If a second number is listed, it is with respect to the de-assertion of the second MWB_CTRL.
3. See the detailed timing diagrams for an example of this command. This timing can only be achieved if the memory write data is unloaded *AFTER* the memory read data is delivered.

Operational Details 4

This chapter covers several important operational details of the XB1 not covered in other chapters. It provides details of the wrapping management for memory read operations, the operation and detailed timing of the memory read buffer, and the operation and detailed timing of the memory write buffer. The PIO write single (X_PIS command) is discussed as well.

4.1 Wrapping Management

The XB1 is designed such that quanta of data returned from any memory fetch is a block. Furthermore, the size of the block is two A_BUS-size words. Systems which require "critical word first" delivery of data to the processor must return the requested word first, followed by the remaining words. Because the connection of the A_BUS is fixed, the only way to vary the delivery of words to the processor (or to I/O) is to perform a swap of the data. The even and odd variants of the X_M[P,I] commands perform just this task.

In the system in which the XB1 is present, the order of delivery of a block of data depends on the address that's requested. To provide data to the processor in the correct order, the XB1 uses the "even" and "odd" memory transfer instructions to realign outgoing data in the necessary fashion. Figure 4-1, "Data Wrapping Management." graphically illustrates data wrapping management.

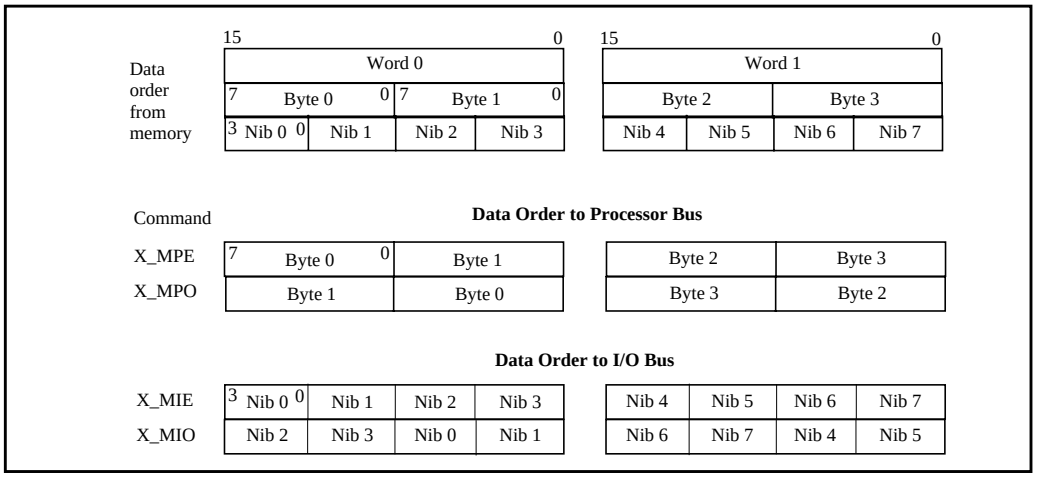

Figure 4-1 Data Wrapping Management

Note that the wrapping management is only applicable to transfers involving the memory read buffer. Writes are strictly sequentially ordered.

4.2 Memory Read Buffer Operation

The memory read buffer stores data coming from memory destined for either the processor or I/O ports. Loading of the buffer is under control of the MRB_CTRL signal and is independent of any unloading command.

Note: The XB1 design allows for overwriting data stored in the memory read buffer. A controller should be able to load data into the buffer, then decide not to use the data and proceed to overwrite the data with new data.

Correct operation to occurs, when the implementation keeps two flags: one for each half of the buffer. If both halves are full, and another MRB_CTRL assertion is detected, the existing data should be overwritten, and the second full flag *must* be cleared. See Figure 4-2 for an example.

It is permissible to issue the overwriting MRB_CTRL coincident with a memory read command. The XB1 should treat this case as though the buffer had been empty and use the bypass mode.

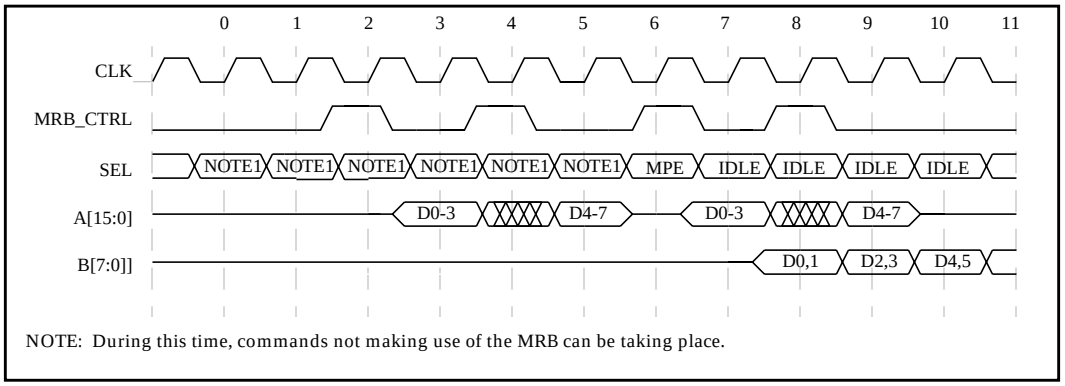

Figure 4-2 MRB Overwrite Timing

In addition to overwriting, it is possible to overlap loading the memory buffer for a subsequent read with the unloading of data from the current read. Figure 4-3 and Figure 4-4 show the most aggressive timing permitted for the MRB_CTRL relative to the command and relative to other MRB_CTRL assertions. The first pair of MRB_CTRL assertions could of course appear earlier in time. It is the controller's responsibility to insure that MRB_CTRL is not asserted in violation of this timing. To summarize: if an X_MP* command is given on cycle N, an

MRB_CTRL which loads new data (as compared with an MRB_CTRL assertion which loads the second half of the active buffer) can be asserted at the earliest in cycle N+3; if an X_MI* command is given in cycle N, an MRB_CTRL which loads new data (as compared with an MRB_CTRL assertion which loads the second half of the active buffer) can be asserted at the earliest in cycle N+5.

Note: Assertions of MRB_CTRL must meet either the overwrite timing, or the pipeline timing. Any other assertion of MRB_CTRL is disallowed and operation is not guaranteed.

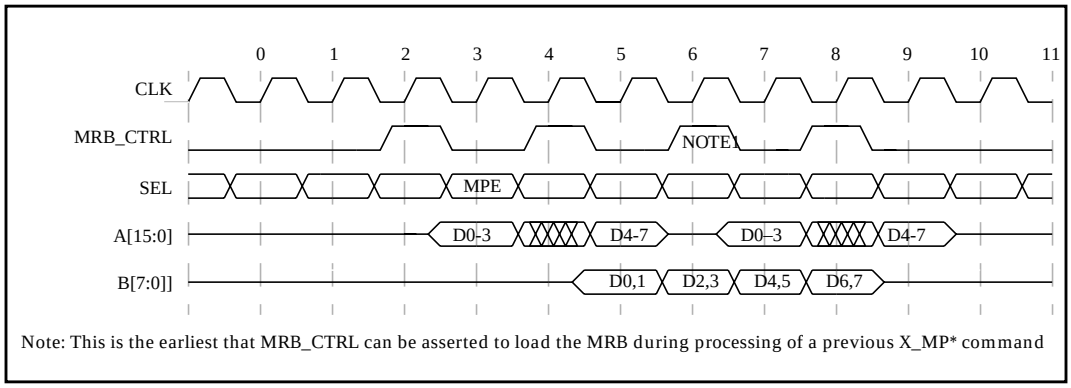

Figure 4-3 MRB MPE Pipelining Timing

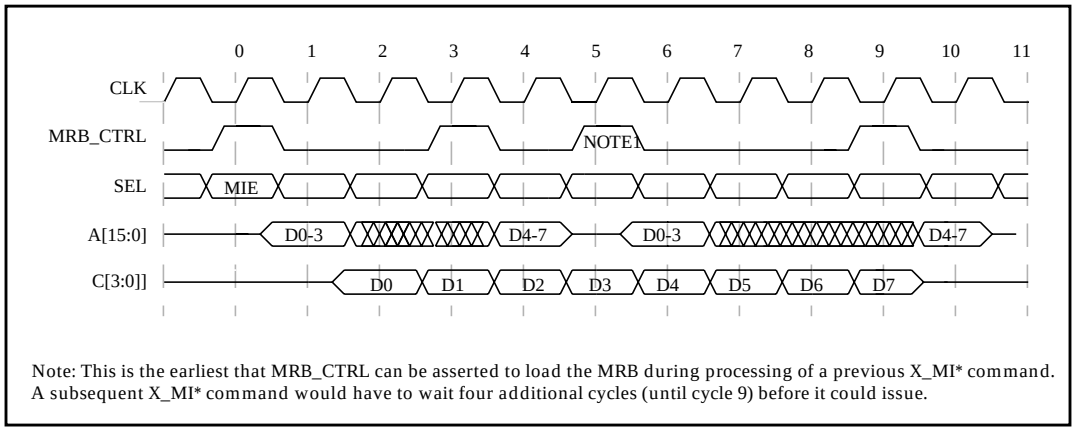

Figure 4-4 MRB MIE Pipelining Timing

4.3 Memory Write Buffer Operation

The memory write buffer provides an isolation between operations on the processor and I/O buses and those on the memory bus. Data is transferred from either the I/O or processor bus into a staging register where it is available for output under control of the MWB_CTRL signal.

Of special note is the handling of the output enable for the A_BUS. As shown in Figure 4-5, "Memory Write Buffer Minimum A_BUS Timing (X_PM)." the assertion of MWB_CTRL at cycle 2 causes the A_BUS data to be driven starting in cycle 3, making it available for sampling on cycle 4. The second MWB_CTRL pulse which occurs at cycle 4 causes the A_BUS data to switch to the other half of the write buffer (MWB). Data will continue to be driven on the A_BUS until one clock cycle after MWB_CTRL is sampled deasserted.

The duration of data hold for the first set of A_BUS write data is controlled by the separation between first assertions of the MWB_CTRL. Data hold for the second set of A_BUS write data is controlled by the length of assertion of the second MWB_CTRL pulse.

Note: Both assertions of MWB_CTRL are pulses and can be of any duration. A one cycle minimum deassertion between pulses is required.

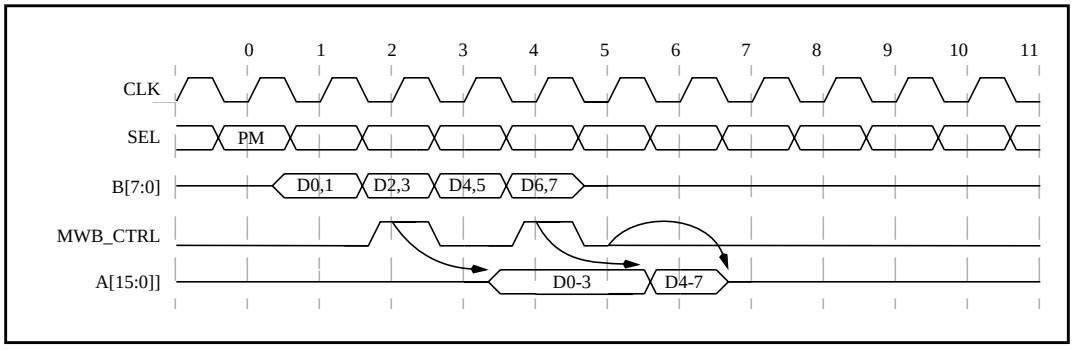

Figure 4-5 Memory Write Buffer Minimum A_BUS Timing (X_PM)

Figure 4-6, "Extending A_BUS Output Enable Timing (X_PM)." illustrates an extension of A_BUS drive time through the use of MWB_CTRL, while Figure 4-7, "Minimum Back-to-Back Processor to Memory Write." illustrates the minimum back-to-back processor-to-memory write timing. The second MWB_CTRL is asserted for two clocks to show symmetrical data hold windows for both sets of data being written to memory. This MWB_CTRL pulse may be one cycle shorter. Moving it a cycle earlier does not improve write performance because of the limitation on issuing back-to-back PM commands.

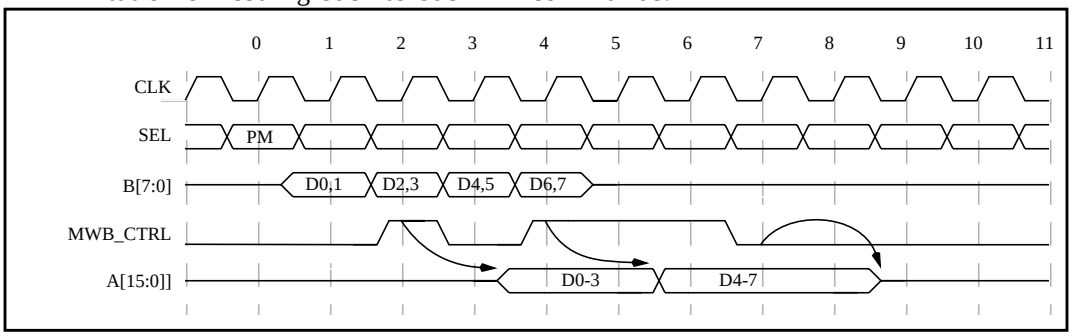

Figure 4-6 Extending A_BUS Output Enable Timing (X_PM)

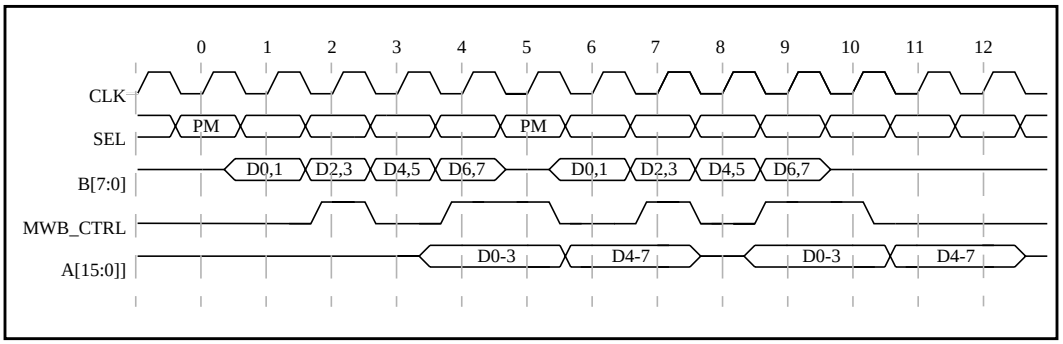

Figure 4-7 Minimum Back-to-Back Processor to Memory Write

4.4 I/O Write Register Operation

A special case is made for single-word-write transfers from the processor bus to the I/O bus. Unlike all other transfers which require a minimum of one dead cycle between transfers, it is possible to stream processor to I/O bus writes with no dead cycles on the C_BUS. Writes have been optimized to improve performance of I/O devices such as graphics adapters. At the boundary between streams of X_PIS commands and any other commands, the C_BUS will go tristate for at least one clock cycle. See Figure 4-8, "Minimum Back-to-Back Processor to I/O Writes." which illustrates three pipelined X_PIS commands followed by an X_MI* command. Note the dead cycle on the C_BUS even though the XB1 is sourcing the data for the X_PIS and X_MI* operations.

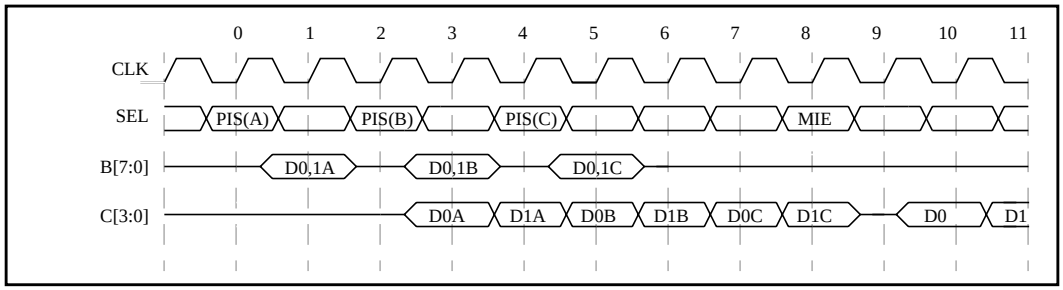

Figure 4-8 Minimum Back-to-Back Processor to I/O Writes

4.5 Timing Diagrams for Command Pairs

This section contains timing diagrams for all of the pair transactions. It is intended to provide examples of the timing listed in Chapter 3, "Command Description." When illustrating any of the memory commands, the even ordering is chosen. The timing is of course identical if the odd ordering is chosen.

4.5.1 X_MI* As the First Transaction

4.5.1.1 X_MI to X_MI Timing

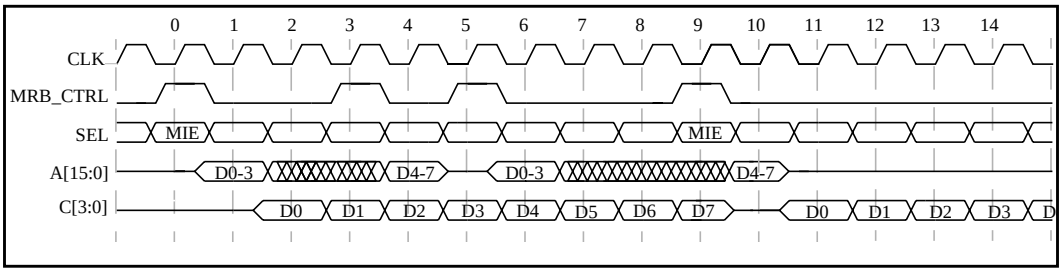

Figure 4-9 MIE-to-MIE Timing

In Figure 4-9, the MRB_CTRLs for the second transfer could have come later while still meeting this timing. Figure 4-10, "MIE-to-MIE Timing When MRB Limited for First Command." illustrates the timing when MRB_CTRL is limited for the first command.

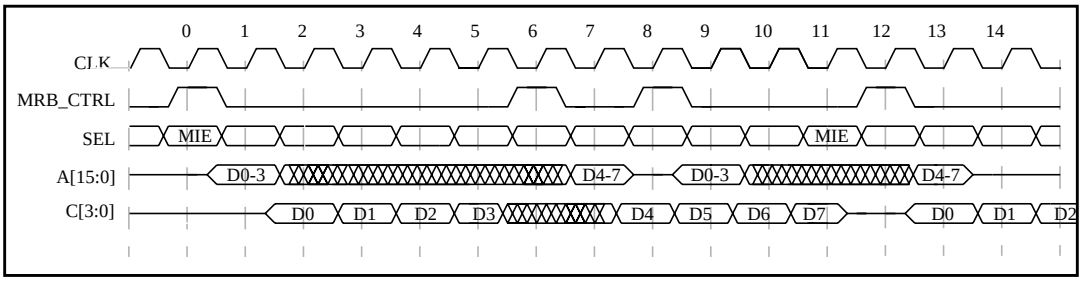

Figure 4-10 MIE-to-MIE Timing When MRB Limited for First Command

4.5.1.2 X_MI to X_MP Timing

Figure 4-11 illustrates the X_MI-to-X_MP timing.

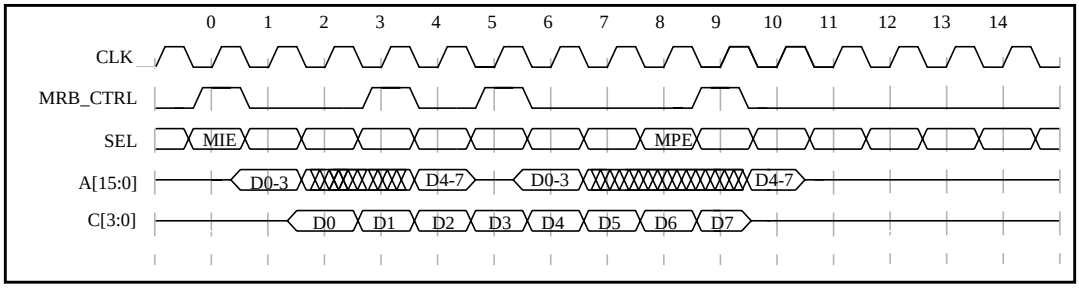

Figure 4-11 MIE-to-MPE Timing

Data for the second transfer is not allowed to appear on the bus until the cycle after the last data word appears on the other bus.

Figure 4-12, "MIE-to-MPE Timing, When MRB Limited for First Command." illustrates the timing when MRB_CTRL is limited for the first command.

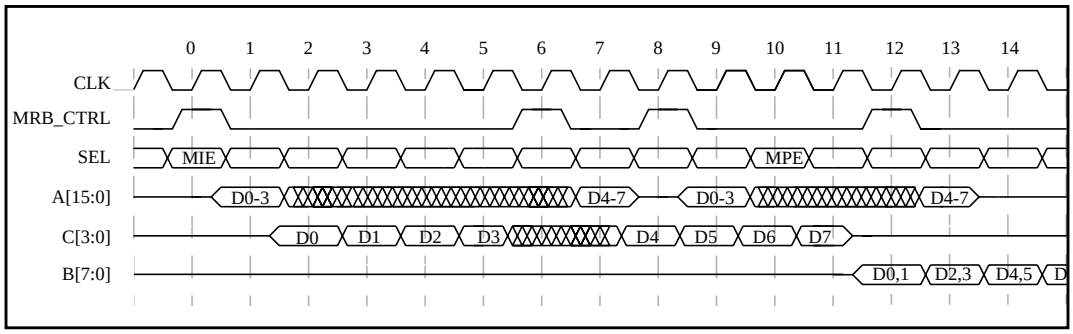

Figure 4-12 MIE-to-MPE Timing, When MRB Limited for First Command

4.5.1.3 X_MI to X_PM Timing

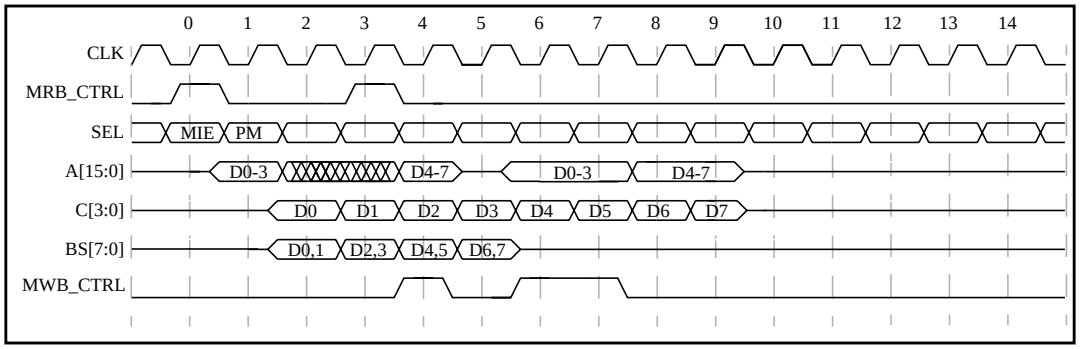

Figure 4-13 MIE-to-PM Timing

In Figure 4-13, "MIE-to-PM Timing.", the data loaded by the X_PM command is not immediately unloaded. Figure 4-14, "MIE-to-PM Timing When MRB Limited for First Command." illustrates that MRB_CTRL does not limit when the X_PM command can issue, but it does affect when the MWB_CTRL can be asserted.

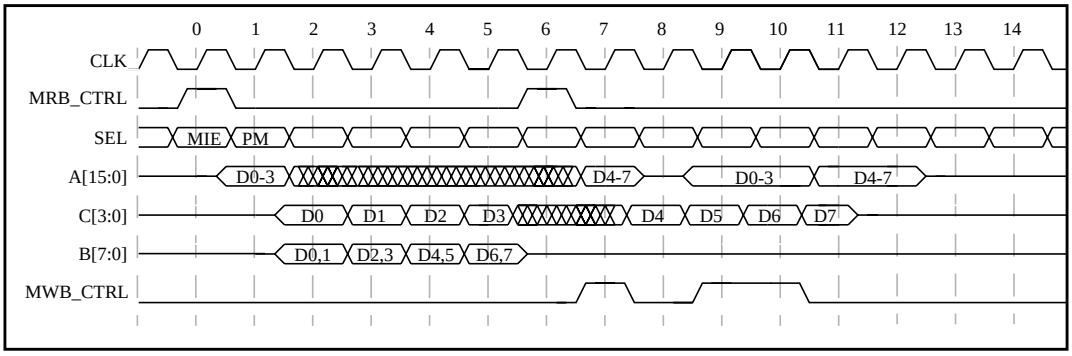

Figure 4-14 MIE-to-PM Timing When MRB Limited for First Command

4.5.1.4 X_MI to X_IM Timing

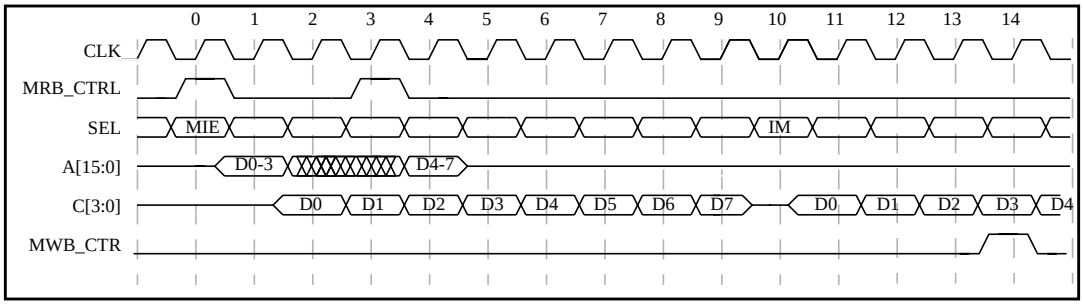

Figure 4-15 MIE-to-IM Timing

Figure 4-15, "MIE-to-IM Timing." illustrates command X_MI-to-X_IM timing. In Figure 4-16, the second MWB_CTRL could be asserted as early as cycle #16.

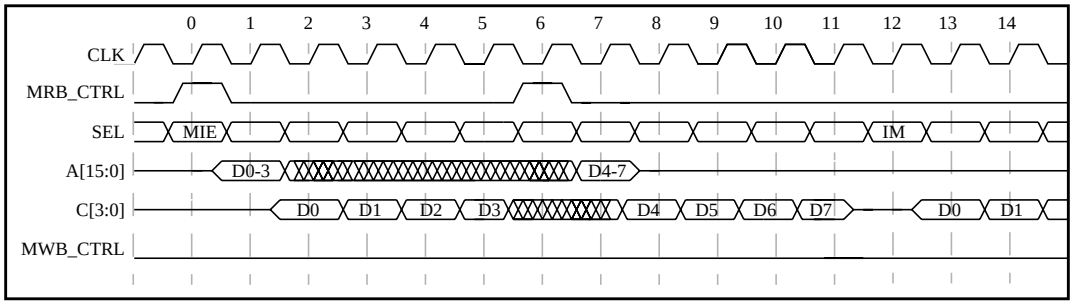

Figure 4-16 MIE-to-IM Timing When MRB Limited for First Command

4.5.1.5 X_MI to X_PIS Timing

In Figure 4-17, "MIE-to-PIS Timing." indicates the command X_MI-to-X_PIS timing, while Figure 4-18, "MIE-to-PIS Timing, When MRB Limited for First Command." is the timing when MRB_CTRL is limited for the first command.

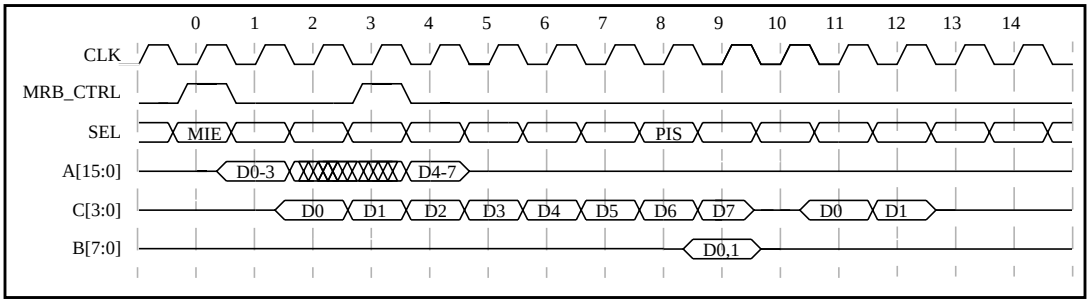

Figure 4-17 MIE-to-PIS Timing

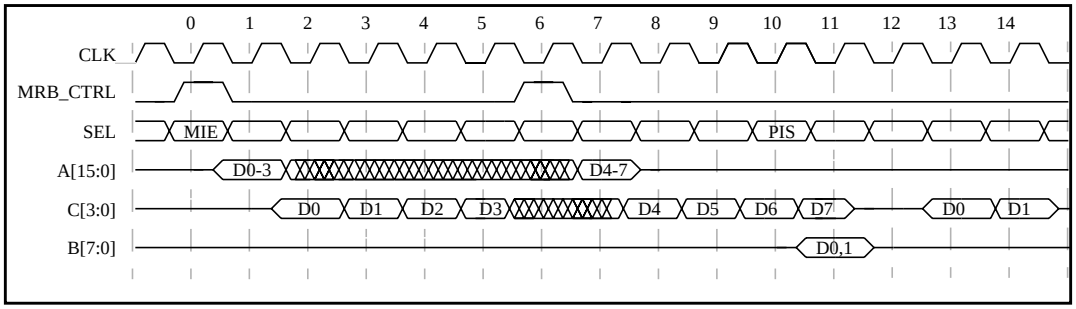

Figure 4-18 MIE-to-PIS Timing, When MRB Limited for First Command

4.5.1.6 X_MI to X_IPS Timing

Figure 4-19, "MIE-to-IPS Timing." illustrates command X_MI-to-X_IPS timing.

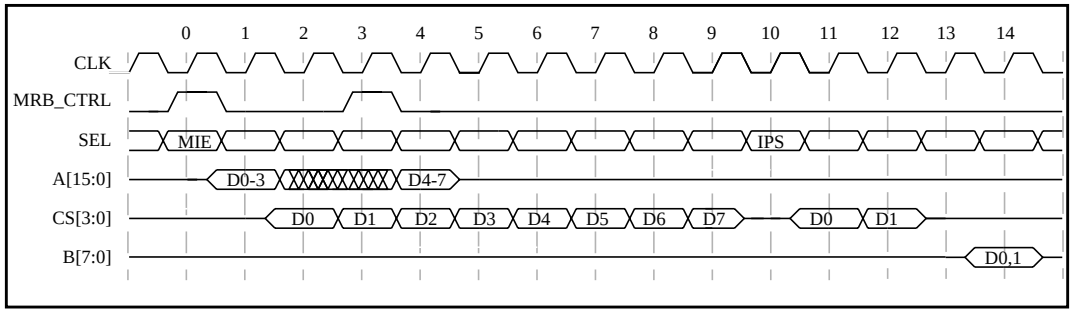

Figure 4-19 MIE-to-IPS Timing

In Figure 4-20, "MIE-to-IPS Timing, When MRB Limited for First Command." the B_BUS is driven on cycle number16.

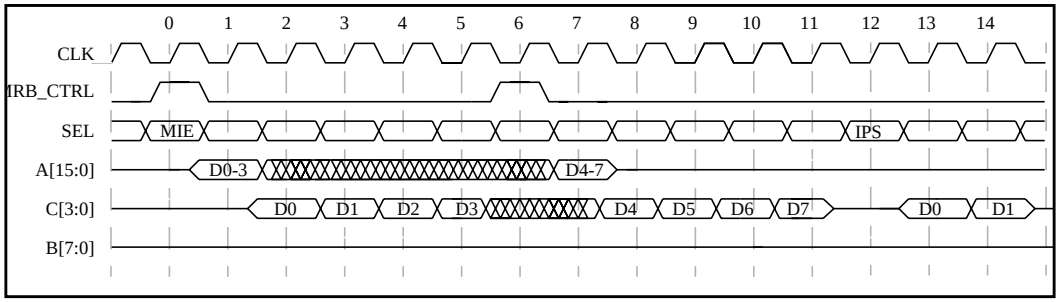

Figure 4-20 MIE-to-IPS Timing, When MRB Limited for First Command

4.5.1.7 X_MI to X_PIB Timing

Figure 4-21 illustrates command X_MI-to-X_PIB timing.

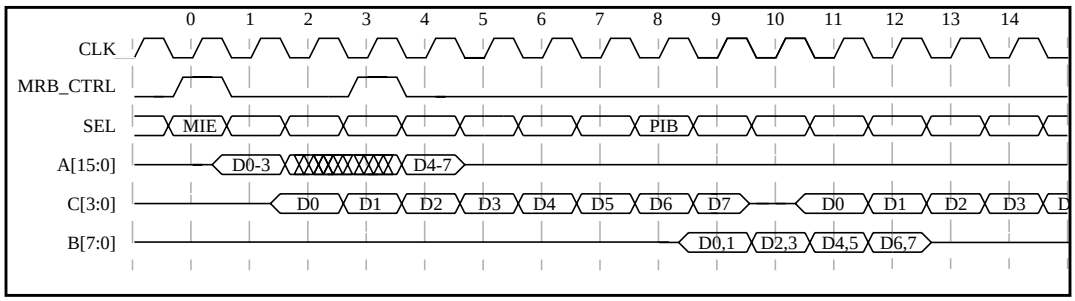

Figure 4-21 MIE-to-PIB Timing

In Figure 4-22, "MIE-to-PIB Timing, When MRB Limited for First Command.", the C_BUS should also be driven on cycles 15 through 20, inclusive.

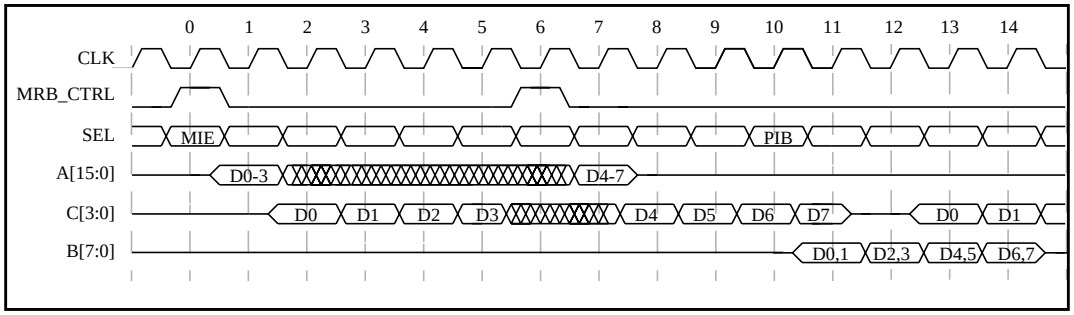

Figure 4-22 MIE-to-PIB Timing, When MRB Limited for First Command

4.5.1.8 X_MI to X_IPB Timing

Figure 4-23, "MIE-to-IPB Timing." illustrates command X_MI-to-X_IPB timing.

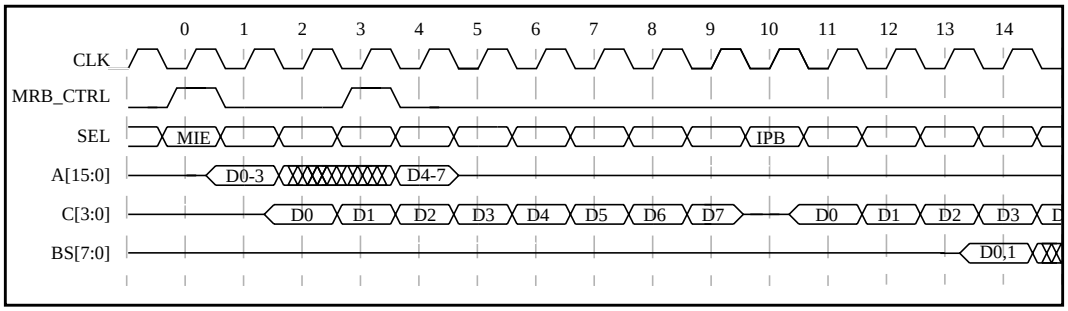

Figure 4-23 MIE-to-IPB Timing

In Figure 4-24, "MIE-to-IPB Timing, When MRB Limited for First Command."the B_BUS sould also be driven cycle 16 through cycle 22, inclusive.

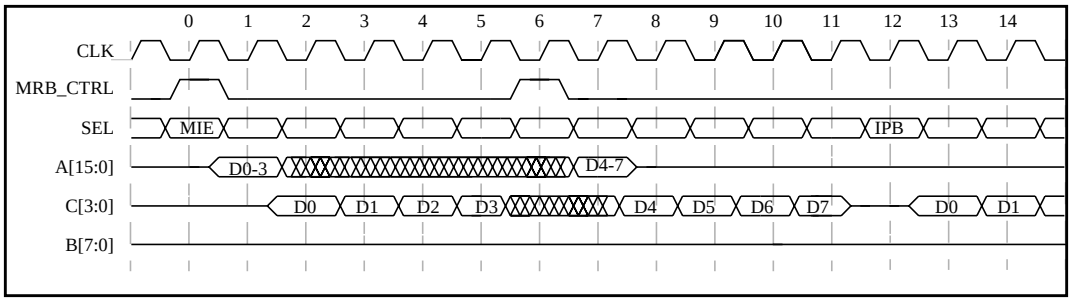

Figure 4-24 MIE-to-IPB Timing, When MRB Limited for First Command

4.5.2 X_MP as the First Transaction*

4.5.2.1 X_MP to X_MI* Timing*

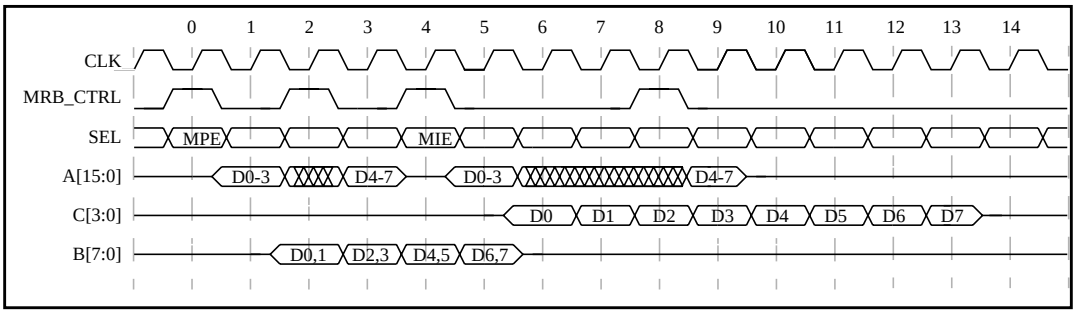

Figure 4-25 MPE-to-MIE Timing

The limiting factor in these cases is data for the second transaction must come at the earliest one cycle after the last B_BUS datum. Therefore, issuing MRB_CTRL any earlier than shown does not change the timing. Figure 4-25, "MPE-to-MIE Timing." illustrates command X_MP*-to-X_MI* timing.

Figure 4-26, "MPE-to-MIE Timing When MRB Limited for First Command." illustrates command MPE-to-MIE timing when MRB is limited for the first command.

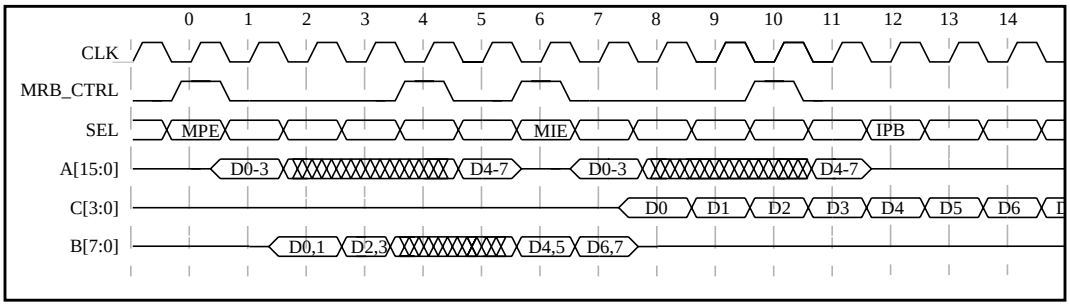

Figure 4-26 MPE-to-MIE Timing When MRB Limited for First Command

4.5.2.2 X_MP to X_MP* Timing*

Figure 4-27, "MPE-to-MPE Timing." illustrates command MPE-to-MPE timing.

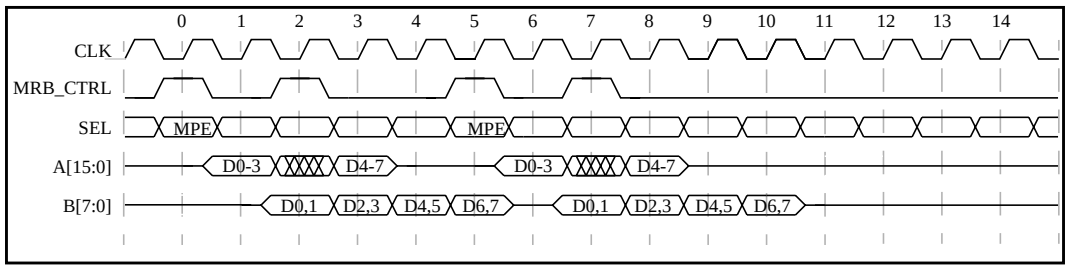

Figure 4-27 MPE-to-MPE Timing

To keep up with the minimum timing, the memory port must be capable of sourcing 66.4 MB/sec/chip. Figure 4-28 illustrates command MPE-to-MPE timing when MRB is limited for the first command.

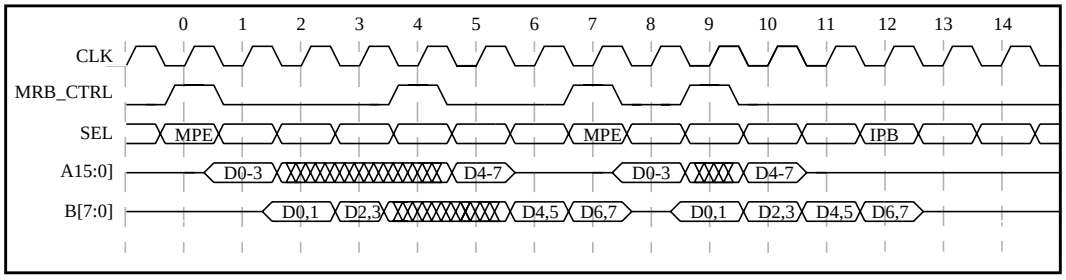

Figure 4-28 MPE-to-MPE Timing When MRB Limited for First Command

4.5.2.3 X_MP* to X_PM Timing

Figure 4-29 illustrates command MPE-to-PM timing and Figure 4-30, "MPE-to-PM Timing, When MRB Limited for First Command." illustrates command MPE-to-PM timing when MRB is limited for the first command.

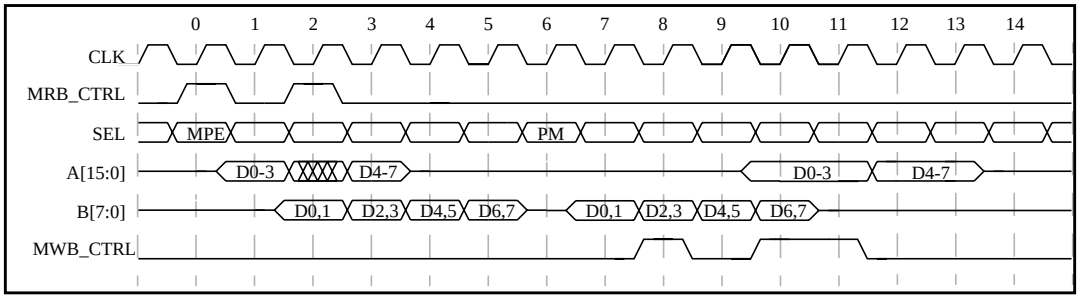

Figure 4-29 MPE-to-PM Timing

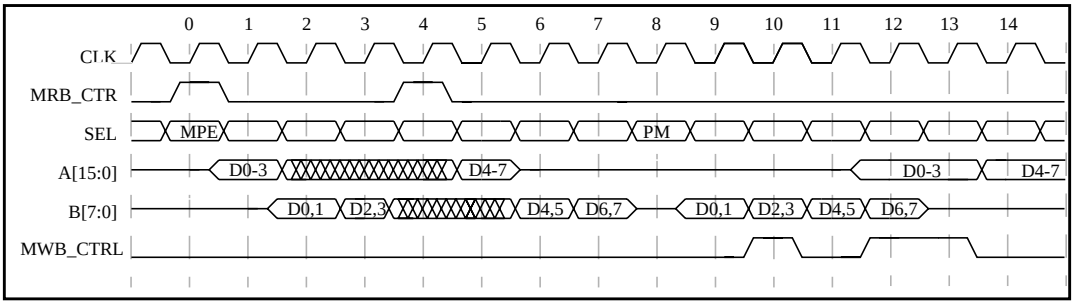

Figure 4-30 MPE-to-PM Timing, When MRB Limited for First Command

4.5.2.4 X_MP to X_IM Timing*

Figure 4-31, "MPE-to-IM Timing." illustrates command MPE-to-IM timing.

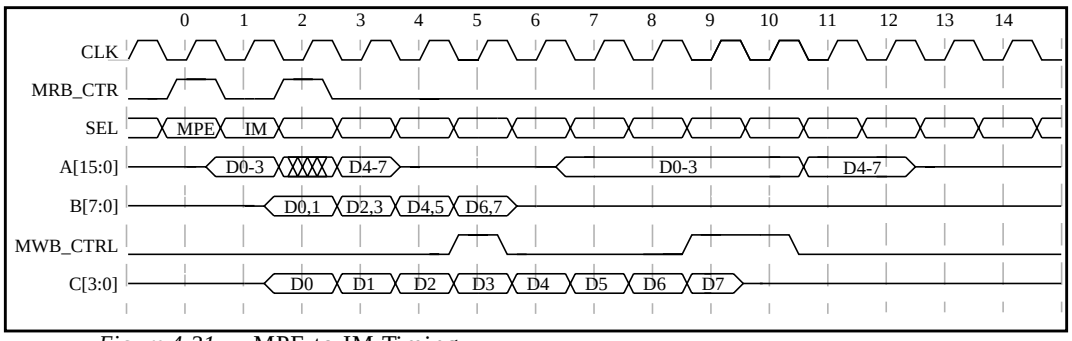

Figure 4-31 MPE-to-IM Timing

The case where the X_MPE command is MRB_CTRL limited is shown below. It is no different than the timing diagram above with respect to when the transfer can be initiated. It is possible to limit when MWB_CTRL may be asserted if the spacing between MRB_CTRLs keeps the A_BUS busy. Figure 4-32 illustrates command MPE-to-IM timing when MRB is limited for the first command.

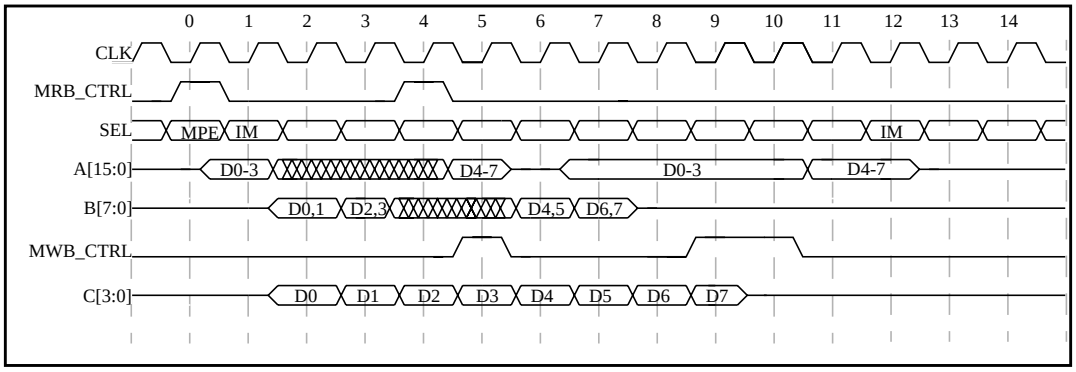

Figure 4-32 MPE-to-IM Timing When MRB Limited for First Command

4.5.2.5 X_MP to X_PIS Timing

Figure 4-33, "MPE-to-PIS Timing." illustrates command MPE-to-PIS timing.

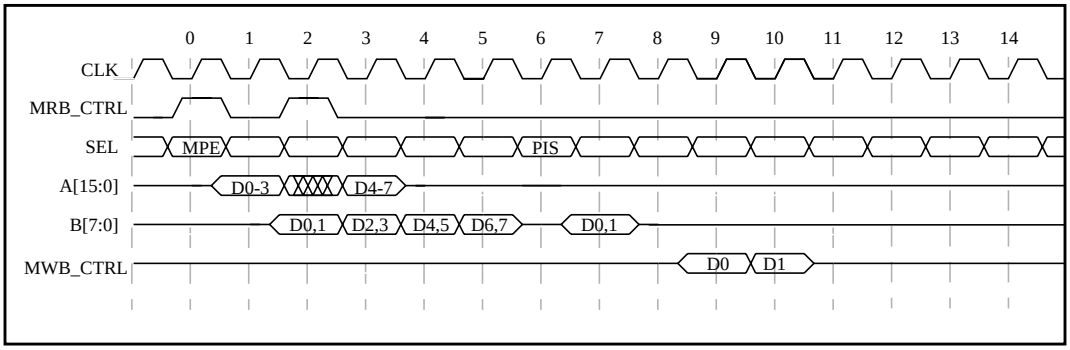

Figure 4-33 MPE-to-PIS Timing

Figure 4-34 illustrates command MPE-to-PIS timing when MRB is limited for the first command.

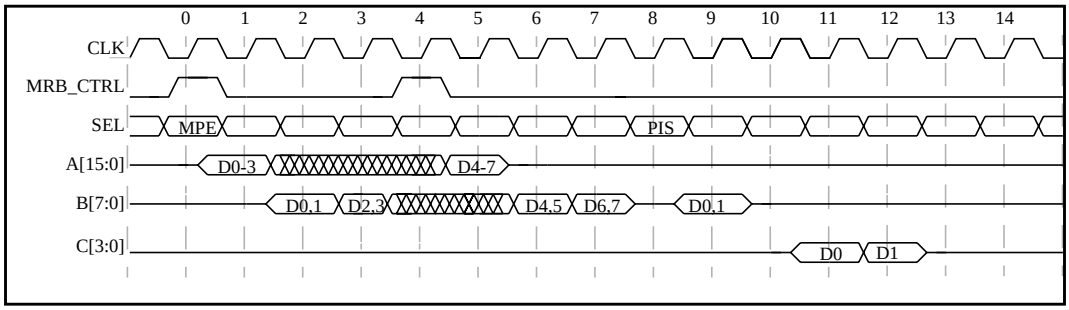

Figure 4-34 MPE-to-PIS Timing When MRB Limited for First Command

4.5.2.6 X_MP to X_IPS Timing

Figure 4-35, "MPE-to-IPS Timing." illustrates command MPE-to-IPS timing while Figure 4-36, "MPE-to-IPS Timing When MRB Limited for First Command." represents command MPE-to-IPS timing when MRB is limited for the first command.

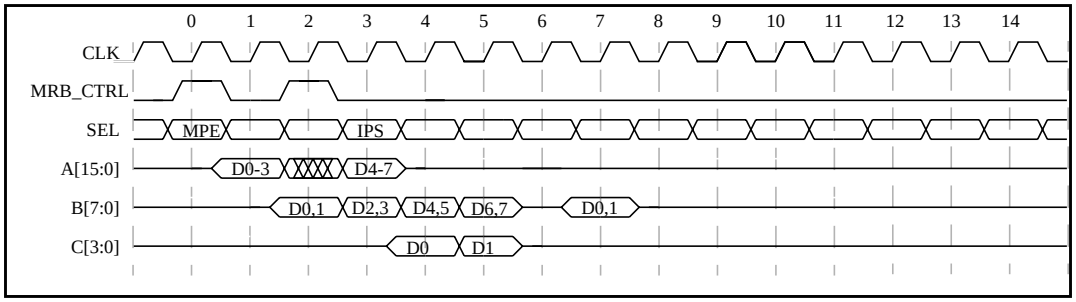

Figure 4-35 MPE-to-IPS Timing

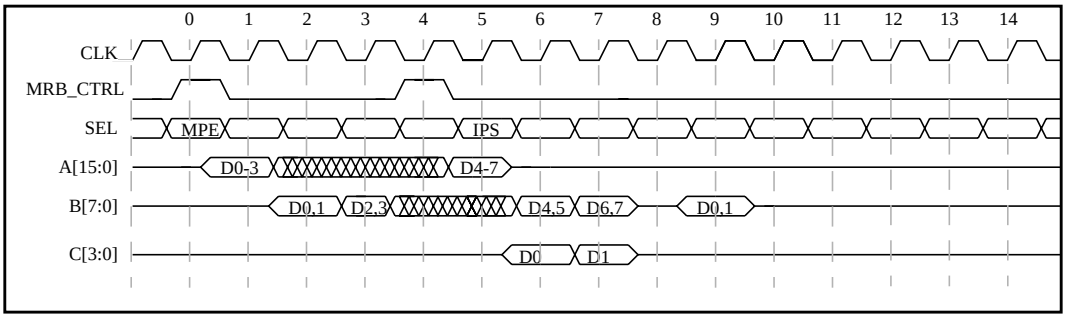

Figure 4-36 MPE-to-IPS Timing When MRB Limited for First Command

4.5.2.7 X_MP to X_PIB Timing

Figure 4-37, "MPE-to-PIB Timing." illustrates command MPE-to-PIB timing and Figure 4-38, "MPE-to-PIB Timing When MRB Limited for First Command." illustrates command MPE-to-PIB timing when MRB is limited for the first command.

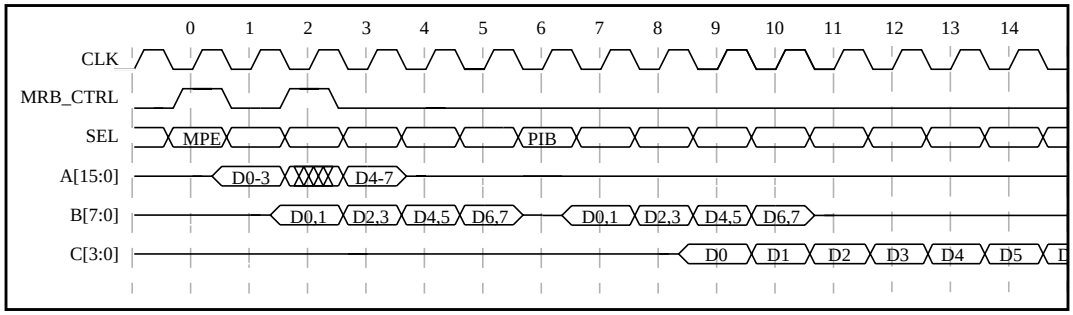

Figure 4-37 MPE-to-PIB Timing

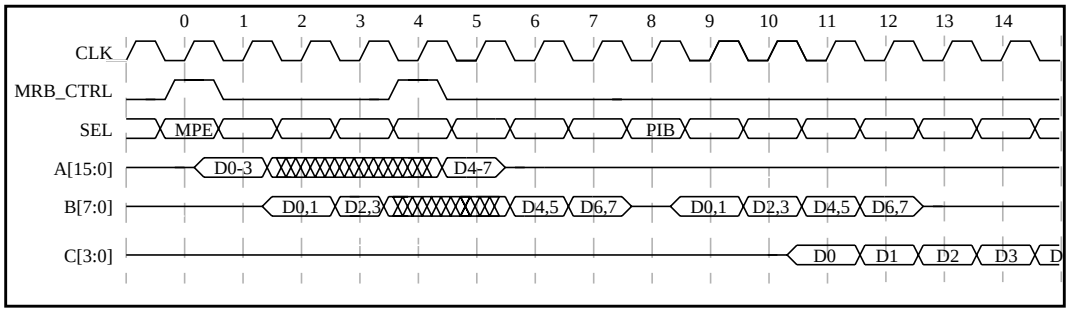

Figure 4-38 MPE-to-PIB Timing When MRB Limited for First Command

4.5.2.8 X_MP to X_IPB Timing

Figure 4-39, "MPE-to-IPB Timing." illustrates command MPE-to-IPB timing.

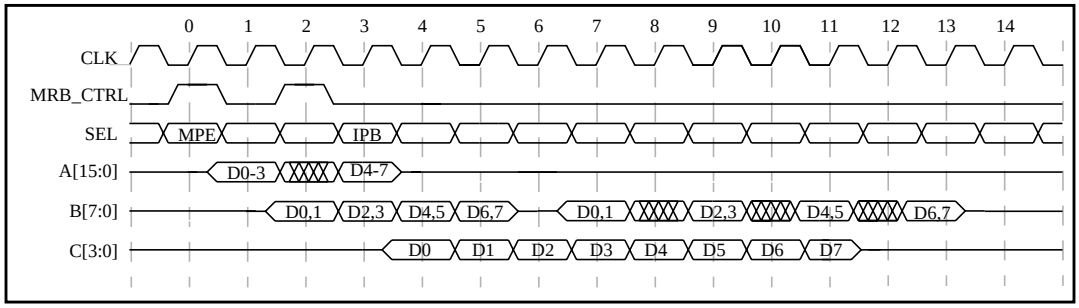

Figure 4-39 MPE-to-IPB Timing

Figure 4-40, "MPE-to-IPB Timing, When MRB Limited for First Command." illustrates command MPE-to-IPB timing when MRB is limited for the first command.

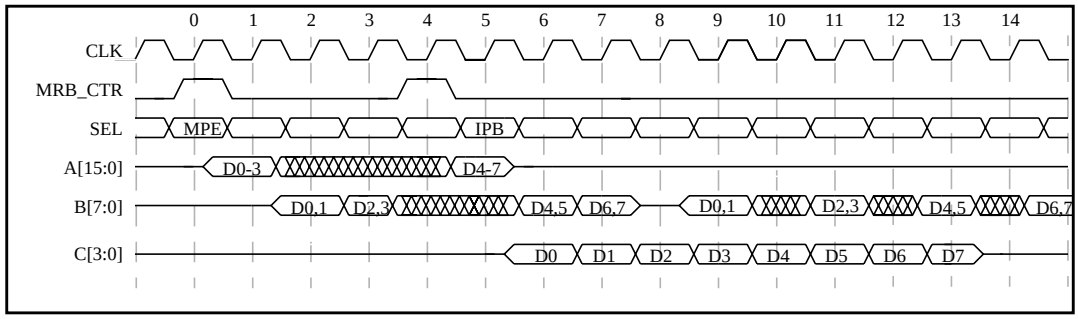

Figure 4-40 MPE-to-IPB Timing, When MRB Limited for First Command

4.5.3 X_PM As the First Transaction

4.5.3.1 X_PM to X_MI* Timing

Figure 4-41, "PM-to-MIE Timing, Write Data Unloaded After Read." illustrates command PM-to-MIE timing, write data unloaded after read.

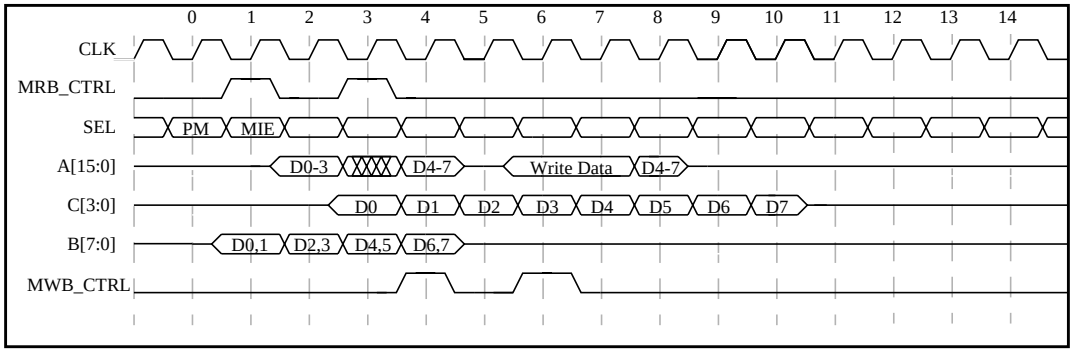

Figure 4-41 PM-to-MIE Timing, Write Data Unloaded After Read

If data is unloaded from the X_PM command first, then the minimum spacing is seven cycles as shown in Figure 4-42, "PM-to-MIE Timing, Write Data Unloaded Before Read."

Note: The timing above is subject to negotiation. If the state machines do not permit simultaneous overlap, the C_BUS data could move out two cycles for a minimum spacing of three cycles rather than one cycle.

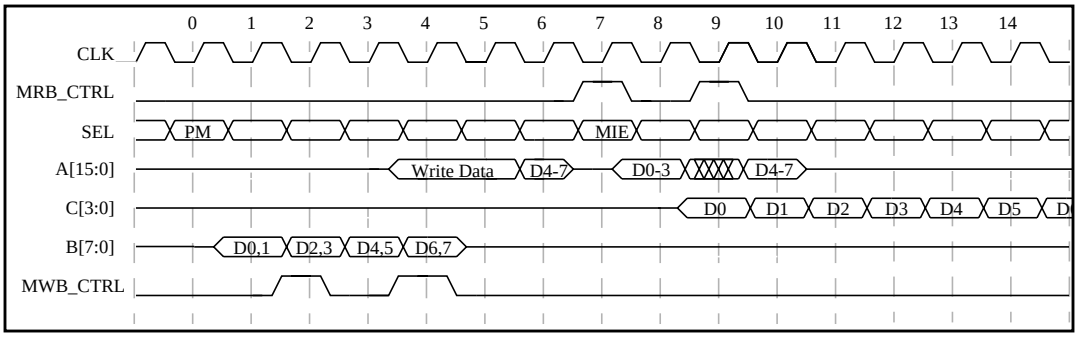

Figure 4-42 PM-to-MIE Timing, Write Data Unloaded Before Read

4.5.3.2 X_PM to X_MP Timing

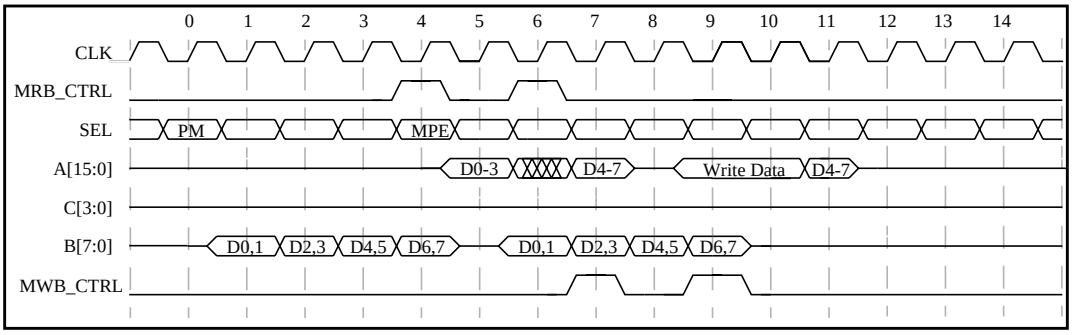

Figure 4-43 PM-to-MPE Timing, Write Data Unloaded After Read

The first set of data on the memory bus for Figure 4-43 is for the read data. The second set of data is the write data. Figure 4-44 illustrates command PM-to-MPE timing, write data unloaded before read.

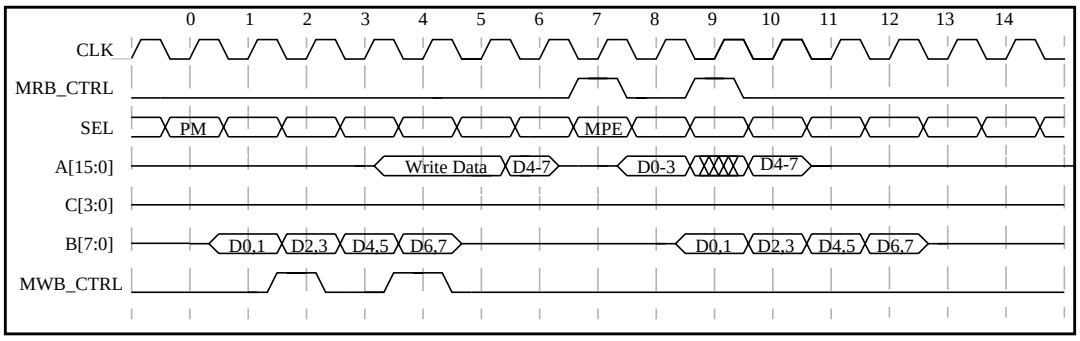

Figure 4-44 PM-to-MPE Timing, Write Data Unloaded Before Read

4.5.3.3 X_PM to X_PM Timing

Figure 4-45, "PM-to-PM Timing." illustrates command PM-to-PM timing.

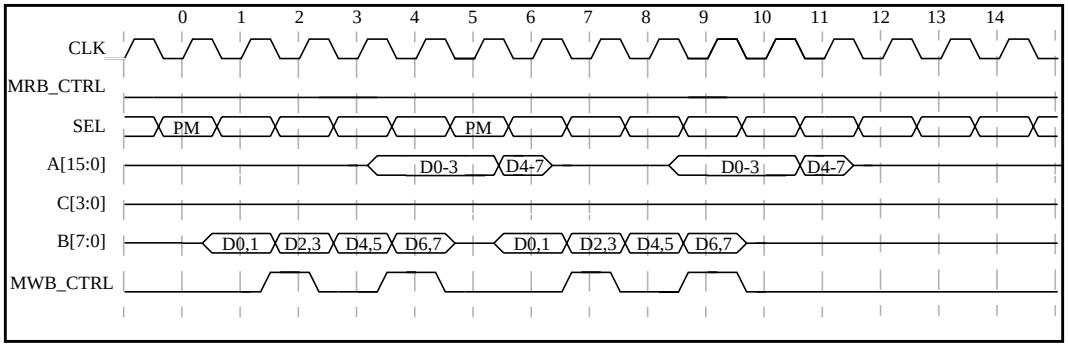

Figure 4-45 PM-to-PM Timing

Figure 4-46, "PM-to-PM Timing, MWB_CTRL Limited." illustrates the timing with MWB_CTRL extended. In order not to overrun the second half of the write buffer, the X_PM command following the first must be no closer than two clocks prior to the deassertion of the second MWB_CTRL for the first transaction. No hardware interlock is provided on the write buffer. It is the controller's responsibility to insure that the write buffer is not overrun.

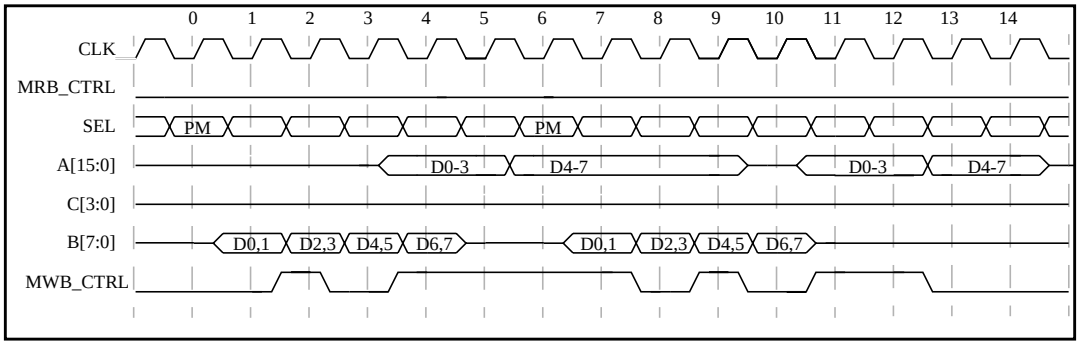

Figure 4-46 PM-to-PM Timing, MWB_CTRL Limited

4.5.3.4 X_PM to X_IM Timing

Figure 4-47, "PM-to-IM Timing." illustrates command PM-to-IM timing.

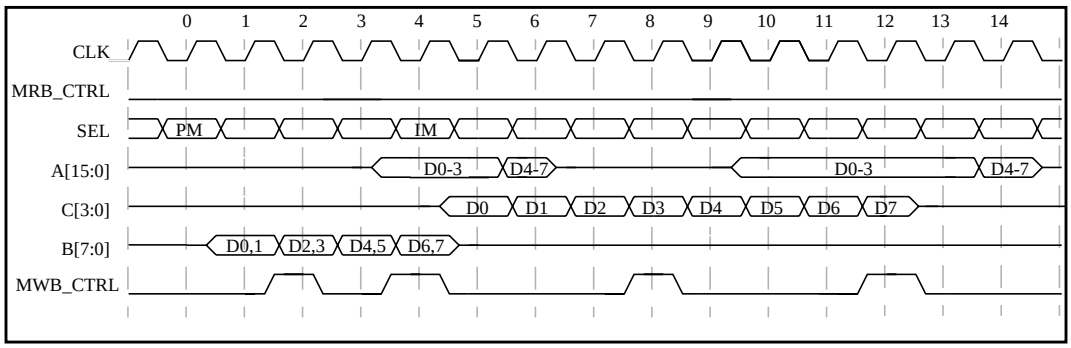

Figure 4-47 PM-to-IM Timing

Figure 4-48, "PM-to-IM Timing, MWB_CTRL Limited." illustrates the timing with MWB_CTRL extended. In order not to overrun the second half of the write buffer, the X_IM command following the first must be no closer than four clocks prior to the deassertion of the second MWB_CTRL for the first transaction. No hardware interlock is provided on the write buffer. It is the controller's responsibility to insure that the write buffer is not overrun.

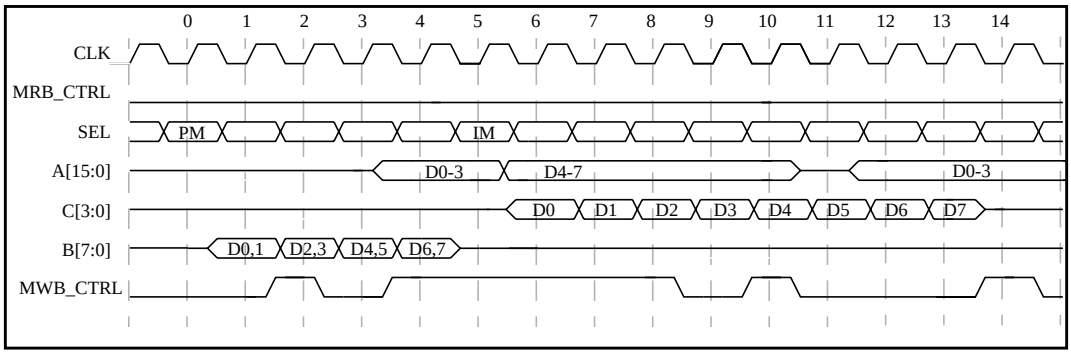

Figure 4-48 PM-to-IM Timing, MWB_CTRL Limited

4.5.3.5 X_PM to X_PIS Timing

This command is not limited due to MWB_CTRL. Figure 4-49 illustrates command PM-to-PIS timing.

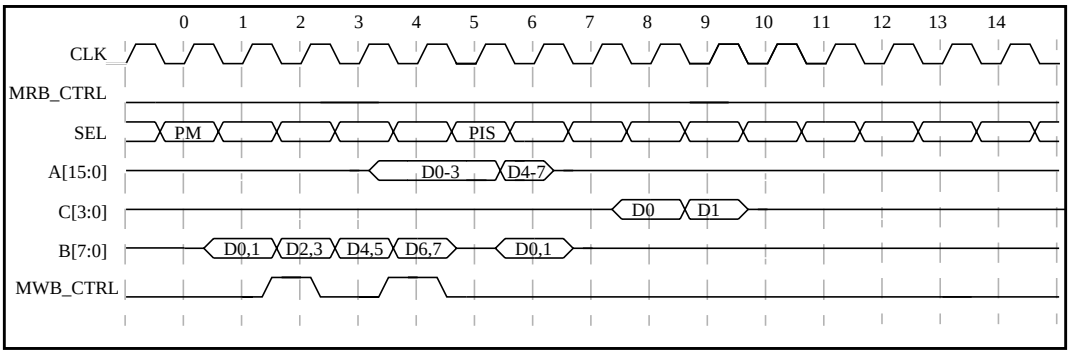

Figure 4-49 PM-to-PIS Timing

4.5.3.6 X_PM to X_IPS Timing

This command is not limited due to MWB_CTRL. Figure 4-50 illustrates command PM-to-IPS timing.

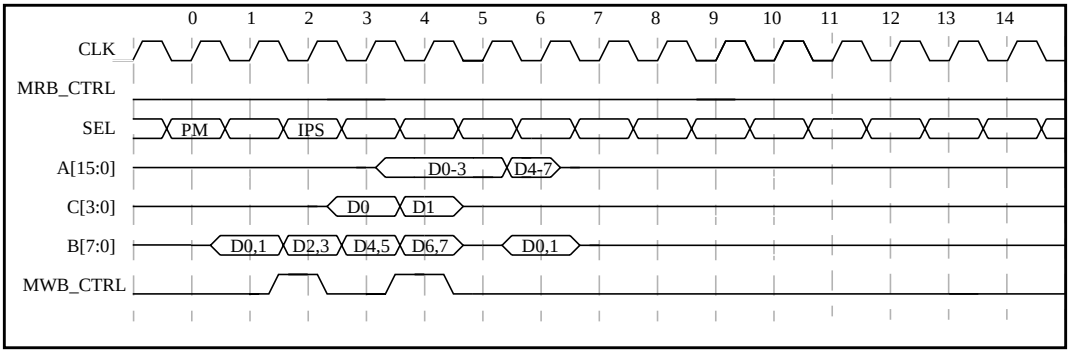

Figure 4-50 PM-to-IPS Timing

4.5.3.7 X_PM to X_PIB Timing

This command is not limited due to MWB_CTRL. Figure 4-51 illustrates command PM-to-PIB timing.

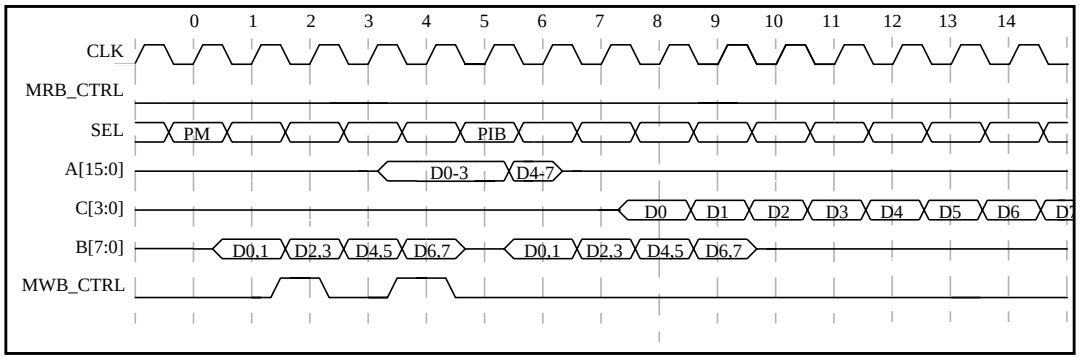

Figure 4-51 PM-to-PIB Timing

4.5.3.8 X_PM to X_IPB Timing

This command is not limited due to MWB_CTRL. Figure 4-52 illustrates command PM-to-IPB timing.

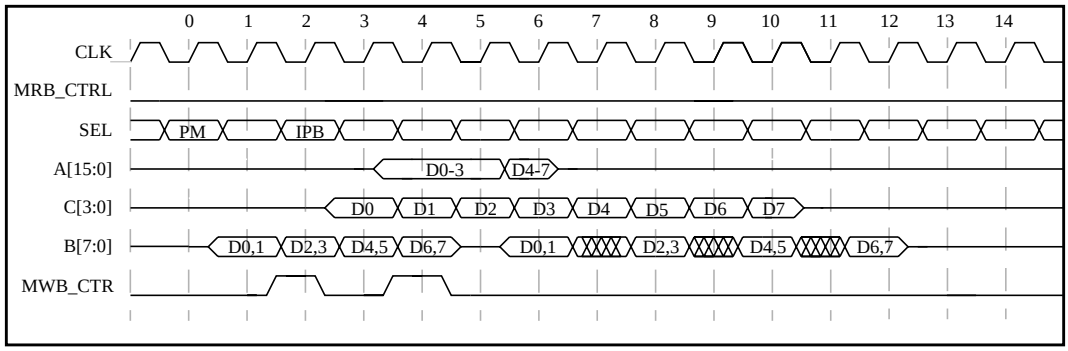

Figure 4-52 PM-to-IPB Timing

4.5.4 X_IM As the First Transaction

4.5.4.1 X_IM to X_MI* Timing

Figure 4-53 illustrates command IM-to-MIE timing, write data unloaded after read.

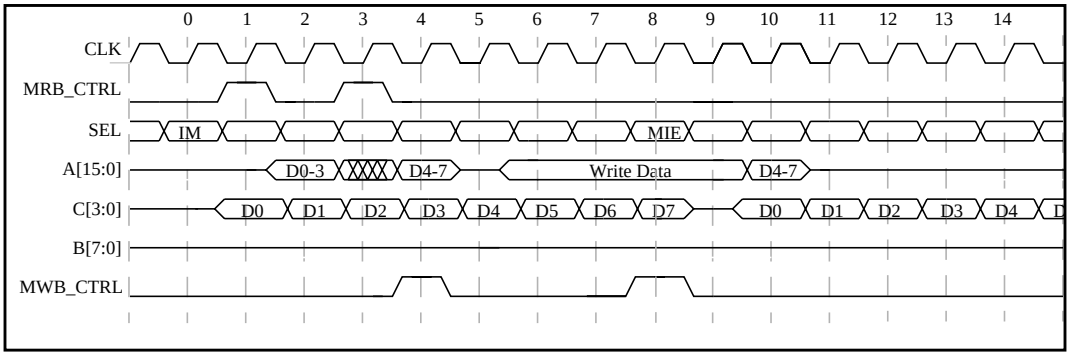

Figure 4-53 IM-to-MIE Timing, Write Data Unloaded After Read

If data is unloaded from the X_IM command first, then the minimum spacing is determined by A_BUS availability and is 11 cycles as illustrated in Figure 4-54.

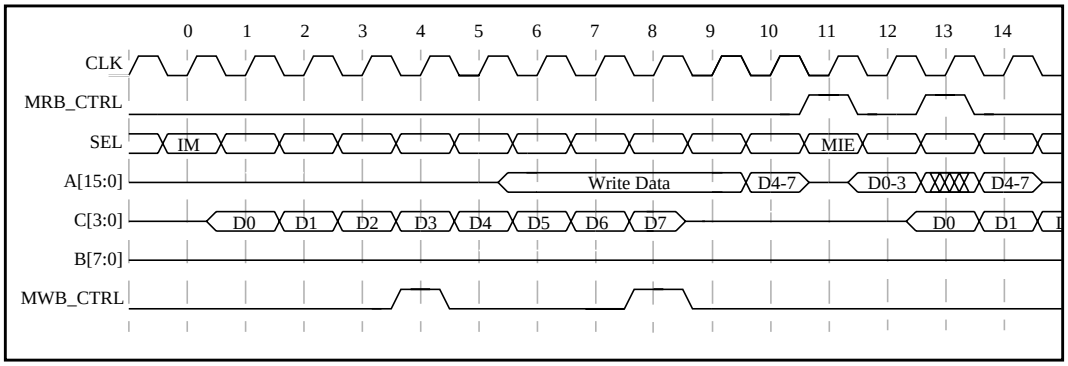

Figure 4-54 PM-to-MIE Timing, Write Data Unloaded Before Read

4.5.4.2 X_IM to X_MP Timing

Figure 4-55 illustrates command IM-to-MPE timing, write data unloaded after read.

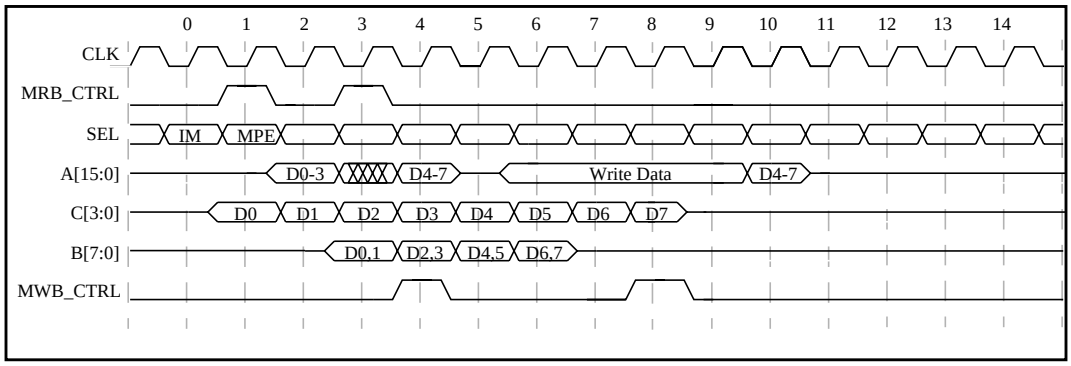

Figure 4-55 IM-to-MPE Timing, Write Data Unloaded After Read

The first set of data on the memory bus for the timing diagram above is for the read data. The second set of data is the write data. In the case that the write data is unloaded before the read, the spacing is eleven cycles minimum as illustrated in Figure 4-56, "IM-to-MPE Timing, Write Data Unloaded Before Read."

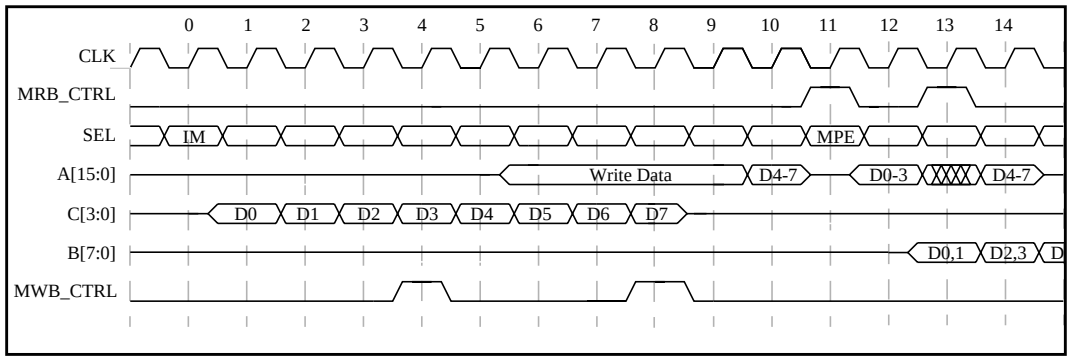

Figure 4-56 IM-to-MPE Timing, Write Data Unloaded Before Read

4.5.4.3 X_IM to X_PM Timing

Figure 4-57 illustrates command IM-to-PM timing.

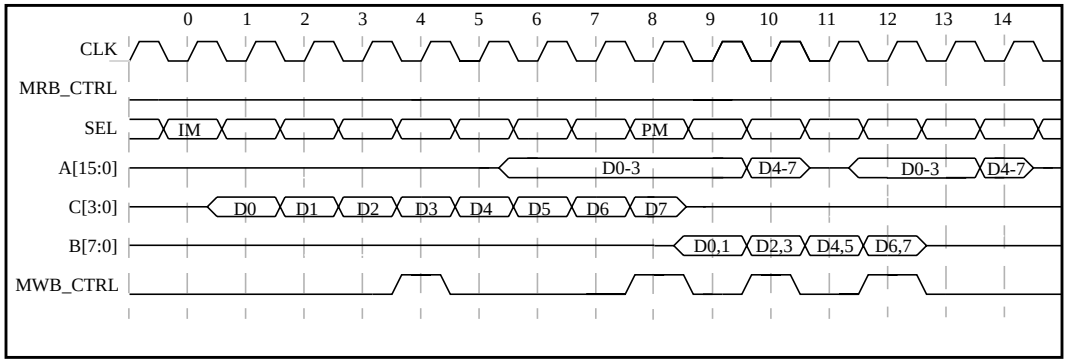

Figure 4-57 IM-to-PM Timing

Figure 4-58, "IM-to-PM Timing, When MWB_CTRL Limited." illustrates the timing with MWB_CTRL extended. In order not to overrun the second half of the write buffer, the X_PM command following the first must be no closer than two clocks prior to the deassertion of the second MWB_CTRL for the first transaction. No hardware interlock is provided on the write buffer. It is the controller's responsibility to insure that the write buffer is not overrun.

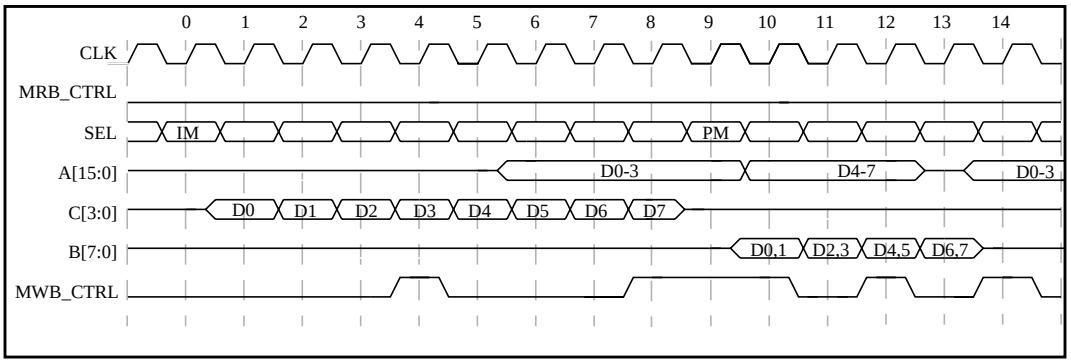

Figure 4-58 IM-to-PM Timing, When MWB_CTRL Limited

4.5.4.4 X_IM to X_IM Timing

Figure 4-59 illustrates command IM-to-IM timing.

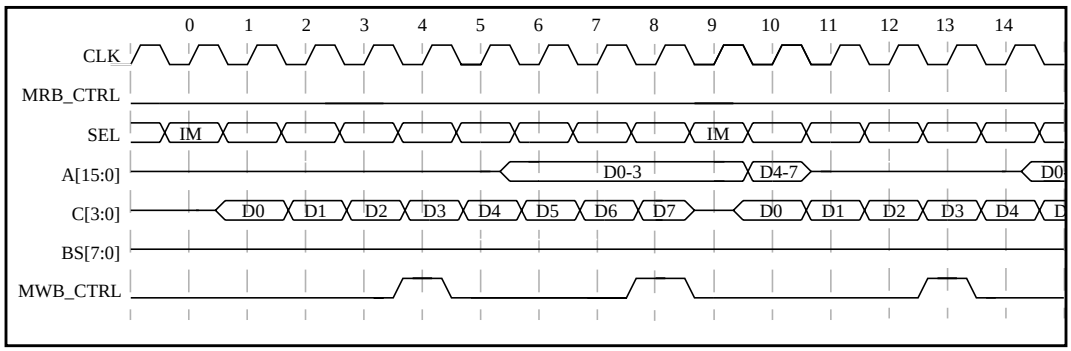

Figure 4-59 IM-to-IM Timing

Figure 4-60, "IM-to-IM Timing, When MWB_CTRL Limited." illustrates the timing with MWB_CTRL extended. In order not to overrun the second half of the write buffer, the X_IM command following the first must be no closer than four clocks prior to the deassertion of the second MWB_CTRL for the first transaction. No hardware interlock is provided on the write buffer. It is the controller's responsibility to insure that the write buffer is not overrun.

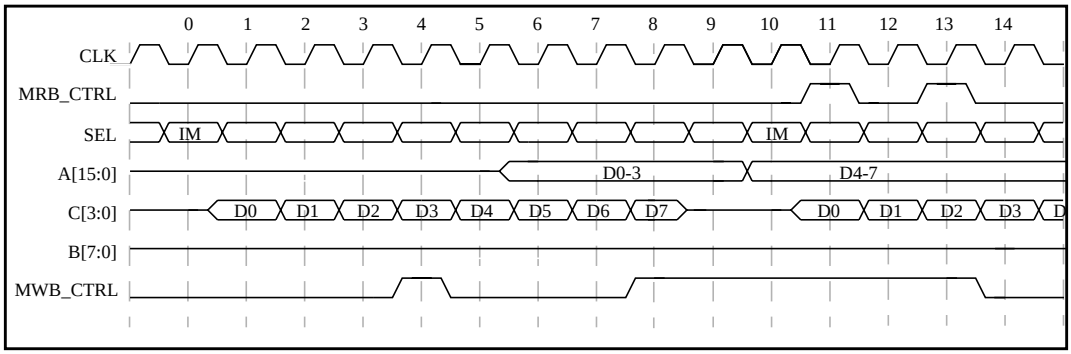

Figure 4-60 IM-to-IM Timing, When MWB_CTRL Limited

4.5.4.5 X_IM to X_PIS Timing

This command is not limited due to MWB_CTRL. Figure 4-61 illustrates command IM-to-PIS timing.

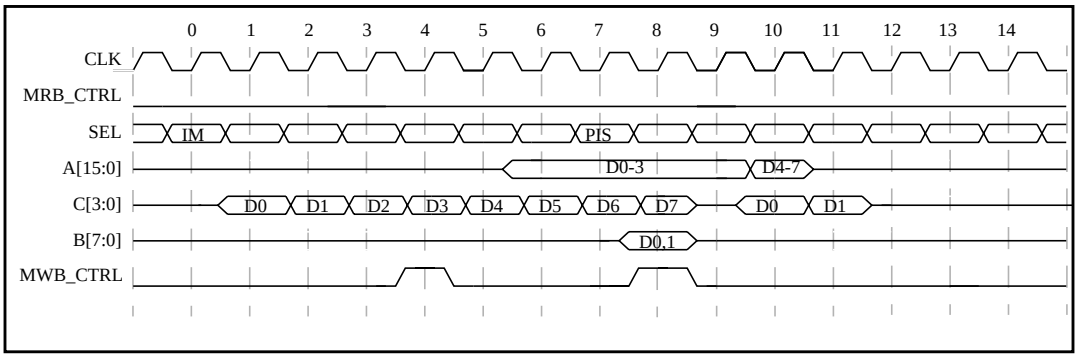

Figure 4-61 IM-to-PIS Timing

4.5.4.6 X_IM to X_IPS Timing

This command is not limited due to MWB_CTRL. Figure 4-62 illustrates command IM-to-IPS timing.

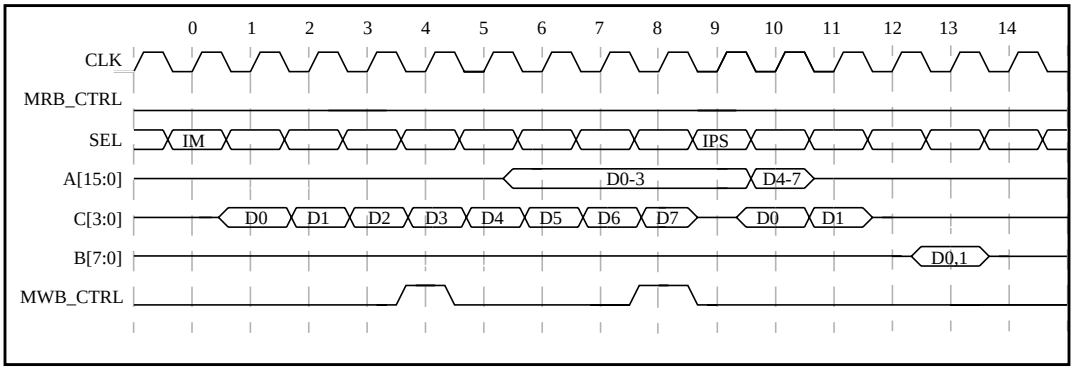

Figure 4-62 IM-to-IPS Timing

4.5.4.7 X_IM to X_PIB Timing

This command is not limited due to MWB_CTRL. Figure 4-63 illustrates command IM-to-PIB timing.

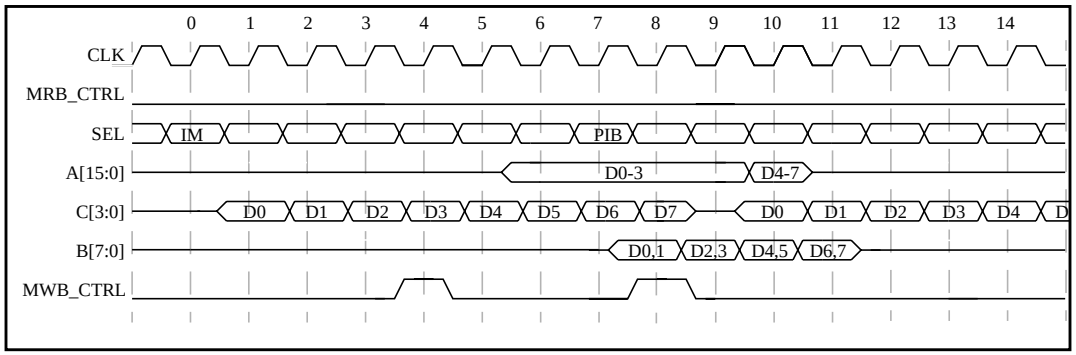

Figure 4-63 IM-to-PIB Timing

4.5.4.8 X_IM to X_IPB Timing

This command is not limited due to MWB_CTRL. Figure 4-64 illustrates command IM-to-IPB timing.

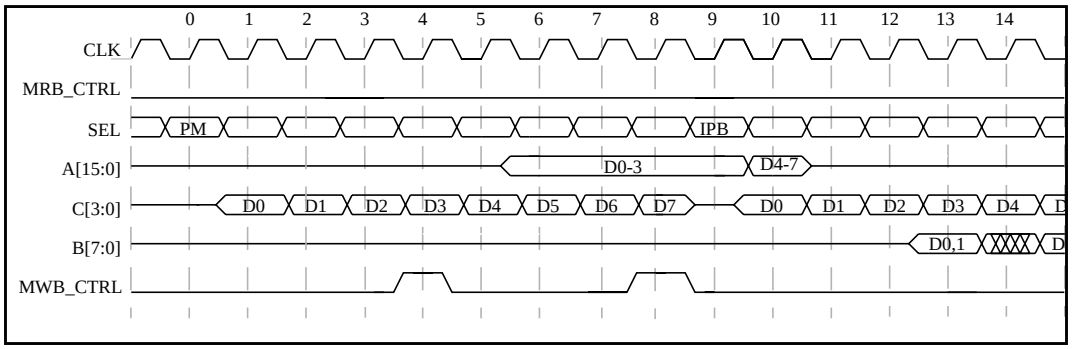

Figure 4-64 IM-to-IPB Timing

Index